# NAVAL POSTGRADUATE SCHOOL
## Monterey, California

# THESIS

AN INTERACTIVE MILITARY PLANNING GAME
FOR THE NAVAL POSTGRADUATE SCHOOL
COMMAND, CONTROL, AND COMMUNICATIONS LABORATORY

by

Scott Cameron Rounce

March 1981

Thesis Advisor: F. Russell Richards

Approved for public release; distribution unlimited.

T199013

# REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

| | |
|---|---|
| 1. REPORT NUMBER | 2. GOVT ACCESSION NO. / 3. RECIPIENT'S CATALOG NUMBER |
| 4. TITLE (and Subtitle): An Interactive Military Planning Game for the Naval Postgraduate School Command, Control, & Communications Laboratory | 5. TYPE OF REPORT & PERIOD COVERED: Master's Thesis (March 1981) |
| | 6. PERFORMING ORG. REPORT NUMBER |
| 7. AUTHOR(s): Scott Cameron Rounce | 8. CONTRACT OR GRANT NUMBER(s) |
| 9. PERFORMING ORGANIZATION NAME AND ADDRESS: Naval Postgraduate School, Monterey, California 93940 | 10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS |
| 11. CONTROLLING OFFICE NAME AND ADDRESS: Naval Postgraduate School, Monterey, California 93940 | 12. REPORT DATE: March 1981 |
| | 13. NUMBER OF PAGES: 193 |
| 14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) | 15. SECURITY CLASS. (of this report): Unclassified |
| | 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE |

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Command, Control, and Communications (C3); wargame; strategic force planning; budgetary planning; resource allocation; TEMPO; interactive computer games

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This thesis implements an adaptation of the TEMPO Military Planning Game for use interactively in the Naval Postgraduate School Command, Control, and Communications (C3) Laboratory. The primary purpose of the game is to provide simulated experience in the allocation of budgetary resources for the development, procurement, and maintenance of military systems

with realistic constraints of limited time and budget, and a degree of uncertainty.

Although the game is resident on the PDP 11/70 computer in the C3 Laboratory, it could easily be transported to any other similar computer which had the same operating system.

Approved for public release; distribution unlimited.

# An Interactive Military Planning Game for the Naval Postgraduate School Command, Control, and Communications Laboratory

by

Scott Cameron Rounce
Captain, United States Air Force
B.A., University of Northern Colorado, 1974
M.A., University of Northern Colorado, 1975
M.B.A., California State University, Dominguez Hills, 1979

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN SYSTEMS TECHNOLOGY - C3

from the

NAVAL POSTGRADUATE SCHOOL
March 1981

[illegible]


## ABSTRACT

This thesis implements an adaptation of the TEMPO Military Planning Game for use interactively in the Naval Postgraduate School Command, Control, and Communications (C3) Laboratory. The primary purpose of the game is to provide simulated experience in the allocation of budgetary resources for the development, procurement, and maintenance of military systems with realistic constraints of limited time and budget, and a degree of uncertainty.

Although the game is resident on the PDP 11/72 computer in the C3 Laboratory, it could be easily transported to any other similar computer system which had the same operating system.

TABLE OF CONTENTS


I. INTRODUCTION _____ 9

II. GENERAL DESCRIPTION_____ 13

III. PLAY OF THE GAME_____ 16

A. OBJECTIVE_____ 16

B. STARTING THE GAME_____ 17

C. ANNUAL PLANNING_____ 18

1. Current Force Status and Data_____ 18

2. Research and Development Data_____ 21

3. Current Budget Data_____ 24

4. Buy Additional Systems_____ 27

5. Modify Systems in Current Inventory_____ 29

6. Scrap Systems in Current Inventory_____ 31

7. Buy Intelligence or Counterintelligence____ 31

8. Perform Trade-off Analyses_____ 34

9. Recovery from Entry Errors_____ 35

10. Submit Completed Planning and Budgeting____ 39

D. WAR_____ 41

E. PENALTIES_____ 44

F. WEAPON SYSTEM EFFECTIVENESS_____ 46

G. DOCUMENTATION_____ 46

H. UMPIRE_____ 48

IV. PHILOSOPHY OF THE GAME_____ 51

A. TEAM ORGANIZATION_____ 51
B. TIME LAG_____ 53
C. ANALYZING THE OPPONENT_____ 55
D. HINTS_____ 56
V. STRUCTURE OF THE GAME_____ 58
A. PROGRAM ARCHITECTURE_____ 58
B. LESSONS LEARNED_____ 62
VI. RECOMMENDATIONS_____ 64
APPENDIX A. USER'S MANUAL_____ 67
COMPUTER PROGRAM _____ 88
BIBLIOGRAPHY _____ 191
INITIAL DISTRIBUTION LIST _____ 192

LIST OF FIGURES


1. Top Level Menu Display_____19
2. Current Force Status Display_____20
3. Available Programs R&D Display_____22
4. Complete R&D Report Display_____23
5. R&D Investment Display_____25
6. Current Budget Display_____26
7. Buy Additional Systems Display_____28
8. Modify Systems Display_____32
9. Scrap Systems Display_____32
10. Buy Intelligence Display_____33
11. Trade-off Analysis Menu Display_____36
12. Trade-off Analysis Report Display_____37
13. Recovery From Entry Error Display_____38
14. Year End Display_____40
15. Sample Year Introductory Report Display_____42
16. Sample Intelligence Report Display_____43
17. Sample War Report Display_____45
18. Utility Discount Scheme_____47
19. Sample Budget Allocation Form_____49
20. Sample Multiple Year Defense Plan Form_____52
21. Time Lag Utility Comparison_____54
22. Top Level Architecture_____58
23. Detailed Program Functional Diagram_____60

ACKNOWLEDGEMENTS

This project would have been virtually impossible to complete without the assistance of Mr. Phil Balma and Dr. Cynthia Irvine of the Computer Science Department. When a program of this size is developed, problems and confusion are normal occurrences. Most of these can be resolved by retracing program steps or trying a different approach, but some of them require the objective eye of a disinterested party. Phil and Cynthia never hesitated to help out when I ran into problems and provide encouragement during the dark hours. I am profoundly grateful for their support.

# I. INTRODUCTION

The Command, Control, and Communications Laboratory (C3 Lab) was conceived and developed to provide future leaders in the field with a testbed to explore ways in which computers and automated communications could improve C3 effectiveness. The project was funded by the Defense Advanced Research Projects Agency (DARPA) and the four services involved in the C3 program, and it was completed early in 1980. The first two classes in the C3 curriculum did not have the opportunity to utilize the laboratory, however, the members of the third class have had extensive experience with the capabilities resident there. Two courses in the curriculum are particularly rooted in the C3 Lab environment. These are Simulation and War Gaming (OS 3655) and C3 Exercise Laboratory (OS 3750). It became apparent during these classes that there was a need for war gaming software to be used by the students for experience in strategic resource planning.

One of the most widely used strategic planning war games is the TEMPO Military Planning Game which was originally developed in the early 1960's by the Technical Military Planning Organization (TEMPO) of General Electric Company in Santa Barbara, California. Since that time it has been used by the Defense Resources Management Course at the Naval

Postgraduate School; the U.S.Army Management School at Ft. Belvoir, Virginia; the Air War College, Air Command and Staff College, and Squadron Officers' School all at Maxwell Air Force Base, Alabama. Additionally, many variations of the game have been developed and used throughout industry and the government.

Play of the game may be approached from several different perspectives ranging from simple diversion to studies of interpersonal behavior. For example, at the Defense Resources Management Course the game is focused almost exclusively on the interpersonal reactions of the players. The game is administered manually with all reports and calculations done in a pencil-and-paper mode. At the Squadron Officers' School, heavy emphasis is placed on planning and teamwork and a mixture of computer input and manual reporting is employed. At Armed Forces Staff College, the objective of the game is quite different from the original, as implemented here, and it offers an entirely different perspective. There the game uses the same type of resource planning methodology, however, one of the more important goals is to woo a developing country through the use of foreign aid grants and military protection.

TEMPO, as implemented in the Naval Postgraduate School C3 Laboratory, is an interactive computer adaptation of the original game. It is entirely automated so the teams need not spend their time on tedious calculations and message

handling exercises. All communication with the game is conducted through computer keyboard displays and all of the timing and scoring functions are executed automatically. By significantly reducing the administrative processes required of the teams, attention can be focused on the coordination and staffing processes which are most representative of the real-world situations the players will encounter.

The primary purpose of TEMPO is to train military leaders in the skills of force planning and resource management under the realistic constraints of time, uncertainty, and a limited budget. These skills are as important in today's international military environment as are those of force employment in theater or strategic conflict.

The manipulation of strategic forces in this game has been greatly simplified to focus the attentions of the players on the force planning and resource allocation aspects of the situation. These aspects include the selection of alternative weapon systems, the maintenance of proper force mixes, the total life cycle costs of the systems, cost effectiveness considerations, long-range planning, and the elements of risk and uncertainty.

The game uses actual weapon systems as the objects of this planning. No attempt has been made to exhaustively analyze the effective utility of a specific red force system against its corresponding blue force system. Opposing

systems were chosen to be "roughly" equivalent in utility, and the names are included solely to stimulate player interest in the game.

There are few features of this version of TEMPO which are unique. It is based primarily on the game as played at Air University in Montgomery, Alabama. Its uniqueness lies in its implementation on the PDP 11/70 computer resident in the C3 Laboratory at the Naval Postgraduate School. This provides players with the capability to interact in real-time with the program through the war-gaming stations in the laboratory.

Appendix A to this thesis is designed to serve as a stand-alone User's Manual for students who will be playing TEMPO in the future. It provides all the necessary background and guidance to accomplish the objectives of the game.

## II. GENERAL DESCRIPTION

TEMPO is played with an umpire and two teams, a red force and a blue force. Both teams start with identical force structures, research and development options, and budgets. That is to say, the costs, utilities, and research and development potentials for corresponding systems on each side are the same even though the systems have different names.

The budget, which varies from year to year, may be spent in any of six ways: (1) operation of forces, (2) acquisition of additional forces, (3) research and development of new systems, (4) modifications to existing weapon systems, (5) intelligence or counterintelligence, and (6) trade-off analyses between systems.

The game spans eight years with the first four years being operational years and the last four years being programmed years. All transactions must be completed within the allotted time, or budget penalties for the ensuing year will be incurred.

The categories of systems which each team has to work with are limited to offensive strategic aircraft, offensive strategic missiles, surface-to-air missiles (aircraft defense), and defensive anti-ballistic missile (ABM)

missiles. Several types of weapon systems are available in each category through the course of the game. System capability is quantified in terms of a standard measure of effectiveness called the "util". Each weapon system is worth a specific number of utils while it is in the active force. The author recognizes that one of the key factors in determining proper force structure is the effectiveness of one's own forces versus the opposition they are most likely to encounter. However, for the purposes of this exercise, utility will be treated as a given, thus allowing concentration on the budgetary aspects of the situation.

The objective of the game is for each team to operate its offensive forces to maximize its net offensive utils while operating a defensive force to minimize the opponent's net offensive utils. The limited budget which each team must work with increases the challenge of this objective. The teams must decide if it is worthwhile to allocate scarce budget dollars to research and development for improvement of the force efficiency. Dollars spent for programs other than strictly procurement of existing systems may result in less than optimum current force structures in order to improve future balance. In addition, teams must decide whether or not to invest in intelligence about their opponent. An analysis option for making cost-benefit tradeoffs is also available to each team.

There are three versions of TEMPO available in the C3 laboratory. The normal version requires four hours to complete the game. This mode allows for extensive planning periods which are desirable for larger teams to derive the full benefit of cooperative staffing in the various areas of concern. The short version requires two hours for completion and has much shorter planning periods. This mode allows the teams to get an exposure to the features of the game under an extreme time constraint. The third version is a demonstration program to allow a single user to examine the features of the game. There are no time constraints involved in the demonstration program.

## III. PLAY OF THE GAME

This chapter describes how to initiate the TEMPO game and outlines the options available to the players as the game progresses.

### A. OBJECTIVE

The objective of the game is for each team to operate its offensive forces to maximize its net offensive utils while operating a defensive force to minimize the opponent's net offensive utils. The formula for determining a team's net offensive utils is:

(total offensive aircraft utils minus opponent's total anti-aircraft utils)

plus

(total offensive missile utils minus opponent's total anti-missile utils)

The net offensive utility resulting from this calculation is compared to the opponent's net offensive utility to determine who wins each year. The teams will be notified at the end of each year whether they won or lost that particular year. The overall winner of the game is the team which has the greater cumulative total of net offensive utils for the eight-year period.

The players should bear in mind that utils are only received for systems which are in the active force. No credit is received for over-defending. That is, it does no good to have more defensive utils in a particular weapon system category than the opponent has offensive utils. And finally, anti-aircraft systems are only effective against aircraft, and anti-ballistic missiles are only effective against offensive missiles.

B. STARTING THE GAME

To begin the game, the teams should login to the computer at their respective stations. The computer directories are 'tempored' and 'tempoblu' with passwords 'tempor' and 'tempob' respectively. The demonstration version of TEMPO is accessed by logging into 'tempdemo' with password 'tempod'.

The first option the teams are offered is whether they want to play the long or short version of the game. TEMPO will not operate properly unless both teams select the same version.

The next few displays the teams receive will be introductory remarks and initial force status. This information is available at any time during the game and, thus need not be recorded.

When the teams have had sufficient time to review the initial data the umpire will signal the beginning of the

game. To initiate play the teams press the carriage return keys at their respective stations and the game commences. TEMPO has a strict timing protocol which it controls automatically from this point on; however, it is important that both teams begin at the same time to ensure that the programs are coordinated. If there is a significant difference in the times that the respective teams initiate play, it is recommended that the game be restarted.

C. ANNUAL PLANNING

The next display to appear on the screens is a menu of options available for the teams to choose from. An example of the menu is shown in figure 1. Note that the time remaining for the year's planning is shown at the top of the display. This countdown will appear on every display until the final plan is submitted for the year. The menu items shown may be reselected as often as desired during each year. These items will be described in detail in the following paragraphs.

1. Current Force Status and Data

This segment of the program provides a summary of all systems in the current operating inventory. An example is shown in figure 2. The display includes number of systems in operation, acquisition cost, operating cost and utility of each system. The number of systems of each type which can be purchased in any one year is also listed.

13:46 YOU HAVE 50 MINUTES LEFT TO COMPLETE THIS YEAR'S PLANNING.

1 - CURRENT FORCE STATUS AND DATA

2 - RESEARCH AND DEVELOPMENT DATA

3 - CURRENT BUDGET DATA

4 - BUY ADDITIONAL SYSTEMS

5 - MODIFY SYSTEMS IN CURRENT INVENTORY

6 - SCRAP SYSTEMS IN CURRENT INVENTORY

7 - BUY INTELLIGENCE DATA OR COUNTERINTELLIGENCE

8 - PERFORM TRADE-OFF ANALYSES BETWEEN PROPOSED SYSTEMS

9 - RECOVERY FROM ENTRY ERRORS

0 - SUBMIT COMPLETED PLANNING AND BUDGETING FOR THIS YEAR

T - TIME UPDATE

PLEASE SELECT A CHOICE FROM THE MENU.

Figure 1. Top Level Menu Display

THIS IS YEAR 1 OF THE GAME.

THE SYSTEMS YOU HAVE IN YOUR PRESENT INVENTORY AND
THEIR ASSOCIATED DATA ARE DETAILED IN THE TABLE BELOW:

| SYSTEM | INVENTORY | AQCOST | OPCOST | UTILS | PURCH LIMIT | TOTAL OPCOST | UTILS |
|---|---|---|---|---|---|---|---|
| B-52G | 40 | 60 | 40 | 25 | 30 | 1600 | 1000 |
| MM-III | 20 | 80 | 60 | 40 | 20 | 1200 | 800 |
| CHAPPARAL | 20 | 100 | 60 | 50 | 30 | 1200 | 1000 |
| SPRINT | 50 | 50 | 40 | 20 | 30 | 2000 | 1000 |



TYPE CARRIAGE RETURN TO RETURN TO MAIN MENU

Figure 2. Current Force Status Display

Finally, the total cost of operations and the total utility for the entire force of each weapon system type is listed. The teams would probably want to review this table at the beginning of each year to ensure that the previous year's buys and scraps were reflected.

2. Research and Development Data

The first display which appears when this option is selected shows which research and development (R&D) programs are available for that particular year. An example is shown in figure 3. The teams have only one year to begin work on a particular program. If they do not select the option they will not have the opportunity the next year. There is one exception to this rule. The R&D required prior to a system modification can be done in any of the first three years.

New research programs are offered only in years one, two, and three of the game. Programs begun in these years may be continued to completion but no new programs will be started beyond year three.

The teams will be asked if they wish to see a complete R&D report for the year. This report is shown in figure 4. It provides estimates of acquisition cost, operation cost, utility and R&D costs for each available program. The teams will be told how long the development will take and what the expected R&D costs for the future years of the program will be. The current year R&D cost is actual. However, the subsequent years' costs, as well as the final utility and

THE SYSTEMS WHICH YOU MAY BEGIN TO R&D THIS YEAR ARE:

OFFENSIVE AVIONICS UPGRADE FOR B-52
FB-111H BOMBER
AIR LAUNCHED CRUISE MISSILE(ALCM)
SPARTAN ABM



DO YOU WISH TO SEE A COMPLETE R&D REPORT FOR THIS YEAR? (y/n)

Figure 3. Available Programs R&D Display

| SYSTEM | AQCOST | OPCOST | UTILS | *****R&D COSTS********* 1ST YR | 2ND YR | 3RD YR | YR TILL AVAIL |
|---|---|---|---|---|---|---|---|
| B-52G (MOD) | 150 | 80 | 80 | 200 | 000 | 000 | 1 |
| FB-111H | 200 | 50 | 200 | 500 | 400 | 700 | 3 |
| ALCM | 100 | 90 | 250 | 400 | 600 | 400 | 3 |
| SPARTAN | 100 | 30 | 100 | 700 | 700 | 000 | 2 |

DO YOU WISH TO INVEST R&D FUNDS IN ANY OF THESE PROJECTS? (y/n)

Figure 4. Complete R&D Report Display

operation cost are only estimates and may increase or decrease depending on how the program progresses.

Finally, in this module, the teams are asked if they wish to spend funds for any of the programs. Note that the full R&D report described above need not be reviewed in order to invest R&D funds. If the team wants to invest funds in R&D a third display appears with a menu of candidate programs. An example of this display is shown in figure 5. The actual and projected R&D costs are again shown along with a status report of ongoing programs. If the team has begun a project in a previous year it is reminded to continue the funding or else jeapordize the delivery schedule. The teams do have the option to cease funding of a program after the first year of R&D has been completed and then resume the development in a later year.

Additional R&D information on a system will be available only if the team initially chooses to pursue the program.

3. Current Budget Data

This module provides each team with the total budget figure for the year, the total cost of operating the force for the year, and a running total of dollars spent up to that point in the year. An example is shown in figure 6. Budget estimates for the next two years are also provided. Actual allocations for future years may vary but the estimates can provide a basis for planning. The

| | R&D COSTS 1ST YEAR | 2ND YEAR | 3RD YEAR | STATUS OF ONGOING PROGRAMS. HAVE COMPLETED: |
|---|---|---|---|---|
| B - UPGRADED B-52G | 200 | 000 | 000 | |
| C - FB-111H BOMBER | 500 | 400 | 700 | |
| G - ALCM | 400 | 600 | 400 | |
| N - SPARTAN ABM | 700 | 700 | 0 | |

NOTE: YOU MUST CONTINUE TO FUND THE PROGRAMS LISTED AS IN PROCESS IN ORDER TO COMPLETE THEM ON SCHEDULE.

SELECT THE SYSTEM YOU WISH TO R&D BY TYPING ITS CORRESPONDING LETTER.

Figure 5. R&D Investment Display

```
YOUR BUDGET FOR YEAR 1 IS $7500 .

BUDGET ESTIMATE FOR NEXT YEAR IS $6800.
BUDGET ESTIMATE FOR THE FOLLOWING YEAR IS $7100.

YOU HAVE ALLOCATED 0 DOLLARS SO FAR THIS YEAR.


KEEP IN MIND THAT THE COST TO OPERATE YOUR CURRENT
FORCES THIS YEAR WILL BE 6000.


PROBABILITY OF WAR THIS YEAR IS 0.25.
ESTIMATED PROBABILITY OF WAR FOR NEXT YEAR IS 0.37.


TYPE CARRIAGE RETURN FOR MAIN MENU
```

Figure 6. Current Budget Display

probabilities of war, as currently estimated, are provided in this module.

One subtlety should be noted here. In order to determine funds available for use for the remainder of the year, total dollars spent must be added to operation costs and the sum subtracted from the total budget for the year.

4. Buy Additional Systems

Selection of this option provides each team with a list of all systems which are available for purchase. See figure 7 for an example. Included in the list are the acquisition costs, utility, and purchase limits for each system. The players simply choose which system they want and the computer will ask how many the team wants to buy. When that input is provided the players receive acknowledgement of the sale. They are told how much was spent and how much of their budget for the year remains. If the team tries to buy more than the purchase limit the computer will provide only the maximum available.

Systems bought in one year will not be operated until the following year. Consequently, no operation costs will be incurred and no utility will be gained until the next year.

A new system is available for purchase as soon as all the R&D has been selected and paid for. A team may acquire units of a new system during the last year of R&D (provided the R&D has been funded), or at any time thereafter. A team

```
THE SYSTEMS AVAILABLE FOR YOUR PURCHASE ARE:

                          AQCOST  UTILS  PURCH LIMIT

A - B-52G BOMBER            60     25       30

F - MINUTEMAN III           80     40       20

J - CHAPPARAL SAM          100     50       30

M - SPRINT ABM              50     20       30

SELECT THE SYSTEM YOU WISH TO BUY BY TYPING ITS CORRESPONDING LETTER.
j
HOW MANY DO YOU WANT?
5
YOU JUST SPENT $500.
YOU NOW HAVE $7000 LEFT THIS YEAR.
(This includes your O&M money)


DO YOU WANT ANY OTHER SYSTEMS? (y/n)n
```

Figure 7. Buy Additional Systems Display

cannot lose its option to buy units of a system once it is in the inventory.

5. Modify Systems in Current Inventory

At certain points in the game the teams will be offered the opportunity to modify or upgrade weapon systems they have in their inventories. These modifications are not possible without developing the technology which goes into them. Therefore, the R&D for the modification must be funded before the modification can take place. This funding is provided through the R&D module described above.

When the modify option is selected the players will be offered candidate systems for upgrade. An example of this display is shown in figure 8. There are two ways a system can be modified. It can be withdrawn from operations for the year, thus incurring no operation costs and providing no utility. Or it can be operated at the unmodified levels until the next year when the modification is complete and it automatically assumes the upgraded levels. The teams must choose how many units they want to modify and how they want to distribute the way they are modified. The computer will provide the necessary advisories.

To purchase upgraded systems directly, (that is, other than upgrading systems already in the inventory) it is still necessary to pay the modification R&D cost (if it has not been paid). The new units can be purchased at a price equal to the procurement cost of the unmodified system plus the

```
THE ONLY SYSTEM WHICH YOU CAN MOD AT THIS POINT IS THE B-52G.

NOTE: YOU MUST FUND THE R&D BEFORE YOU CAN MODIFY.


MOD COSTS FOR THE B52-G ARE 90 PER UNIT.
DO YOU WISH TO UPGRADE ANY OF YOUR B-52's? (y/n)
y
YOU HAVE 40 UNITS AVAILABLE FOR MODIFICATION.
HOW MANY DO YOU WANT?
10
YOU JUST SPENT $900.
YOU NOW HAVE $6400 LEFT THIS YEAR.
(This includes your O&M money)


YOU MAY CHOOSE HOW MANY OF THE UNITS BEING
MODIFIED YOU WANT LEFT IN OPERATION WITH THE
UNMODIFIED UTILITY RATE AND OPERATION COST.
HOW MANY DO YOU WANT?
5
```

Figure 8. Modify Systems Display

modification cost. This option will automatically be made available to the teams through the BUY module when the modification R&D is completed.

Modifications may be made at any rate not exceeding the purchase limit of the original system. R&D for the modifications may be performed at any time during the first three years of the game.

6. Scrap Systems in Current Inventory

It may be necessary at certain junctures of the game to get rid of older, less-efficient systems in order to free up their operations costs for other purposes. This can be done by selecting the SCRAP menu choice. A complete list of systems in the current inventory is displayed and the team is asked which system it would like to scrap (see figure 9). It is then asked how many units it would like to take out of operation. The computer acknowledges the entry. Force units scrapped in one year will remain in the inventory until the following year.

7. Buy Intelligence Data or Counterintelligence

Selection of this module provides the team with a menu of ten options for intelligence gathering (see figure 10). For $100 each the team can get a complete report of the opponent's:

a) current force structure of offensive forces, or

b) current force structure of defensive forces, or

```
THE SYSTEMS AVAILABLE FOR YOU TO SCRAP ARE:

                      OPCOST  UTILS

A - B-52G BOMBER        40      25

E - MINUTEMAN III       60      40

J - CHAPPARAL SAM       60      50

M - SPRINT ABM          40      20

SELECT THE SYSTEM YOU WISH TO SCRAP BY TYPING ITS CORRESPONDING LETTER.
e
YOU PRESENTLY HAVE 20 MINUTEMAN III's.
HOW MANY DO YOU WISH TO SCRAP?
5
NEXT YEAR'S INVENTORY WILL BE 5 LESS.

DO YOU WANT TO SCRAP ANY OTHER SYSTEMS? (y/n)
n
```

Figure 9. Scrap Systems Display

```
WHAT TYPE OF INTELLIGENCE DO YOU WISH TO OBTAIN?

YOUR CHOICES ARE:

        1 - CURRENT FORCE STRUCTURE OF OFFENSIVE FORCES

        2 - CURRENT FORCE STRUCTURE OF DEFENSIVE FORCES

        3 - CURRENT R&D PROGRAMS FOR OFFENSIVE FORCES

        4 - CURRENT R&D PROGRAMS FOR DEFENSIVE FORCES

        5 - OFFENSIVE FORCE COUNTERINTELLIGENCE

        6 - DEFENSIVE FORCE COUNTERINTELLIGENCE

        7 - OFFENSIVE R&D COUNTERINTELLIGENCE

        8 - OFFENSIVE R&D COUNTERINTELLIGENCE

        9 - NO FURTHER INTELLIGENCE INFO DESIRED

        0 - CANCEL ALL INTELLIGENCE REQUESTS ENTERED THIS YEAR


PLEASE SELECT 1-0
1

YOU HAVE JUST SPENT $100.YOU NOW HAVE $6300 LEFT THIS YEAR.
(This includes your O&M money)
DO YOU WANT ANY MORE INTEL? (choose 1-0)
9
```

Figure 10. Buy Intelligence Display

c) current R&D programs for offensive systems, or

d) current R&D programs for defensive systems.

Intelligence reports on force structure do not provide an exact count of enemy systems. Rather, they present a ten-unit range for each system, e.g."SYSTEM-A has 20-29 units" is the report if the opponent has 22 units.

R&D intelligence reports list all programs for which R&D funds were expended that year by the opponent. This will not indicate if he has started a program in an earlier year and not funded it during the current year.

Either team may purchase counterintelligence in any of the four areas described above for $50 per category. Counterintelligence reports inform the team if the enemy has purchased intelligence in that category during the year.

The intelligence menu provides the option to cancel all orders for intelligence and start over again for the year.

If an intelligence report is requested it will be presented automatically at the beginning of the following year's planning period.

Intelligence reports cannot be obtained in years five through eight.

8. Perform Trade-Off Analyses

This module may be selected any time during years one through four to assist in making decisions as to which systems to buy. The analysis branch provides a complete list of systems which are, or could be, available at that

time for procurement or R&D. The menu is shown in figure 11. The team is asked to select two systems to be compared and an immediate analysis is presented as shown in figure 12. The report includes year-by-year life-cycle-cost breakouts for both selected systems and yearly utility figures. Totals for both of these parameters are tabulated and used to compute a util/dollar cost effectiveness factor. This value is simply the total life cycle utility divided by the total life-cycle cost.

The teams will not be prevented from performing trade-offs using systems they may have chosen not to R&D. This kind of analysis is not very useful, however, since the preferred option may not be possible to obtain.

The cost of analysis is $25 for each trade-off performed.

9. Recovery From Entry Errors

This module is merely a tutorial to explain how to change entries the teams may have made and then later reconsidered. It is not possible to recover from every entry error, particularly in the R&D module. A team cannot "un-start" development of a new technological program until the next year. For this reason it is strongly recommended that the teams carefully plan their expenditures and programs for the year before they enter the data. Figure 13 shows the text of the tutorial.

WELCOME TO THE ANALYSIS BRANCH.

THIS SUBPROGRAM ENABLES YOU TO COMPARE
TWO SYSTEMS TO DETERMINE WHICH IS THE
MOST COST EFFECTIVE PURCHASE.

THE CANDIDATE SYSTEMS FOR ANALYSIS WILL BE CHOSEN FROM THE
FOLLOWING LIST:

A - B-52G BOMBER
B - UPGRADED B-52G
C - FB-111H BOMBER
E - MINUTEMAN III ICBM
G - AIR LAUNCHED CRUISE MISSILE
J - CHAPPARAL SAM
M - SPRINT ABM
N - SPARTAN ABM

PLEASE SELECT THE LETTER OF THE FIRST SYSTEM IN THE COMPARISON.
a
AND THE LETTER OF THE SECOND SYSTEM IN THE COMPARISON.
b

Figure 11. Trade-off Analysis Menu Display

THE B-52G IS CURRENTLY OPERATIONAL. SINCE THE FOCUS OF THE GAME IS HOW TO BEST UTILIZE THE AVAILABLE BUDGET RESOURCES FOR THE REMAINDER OF THE GAME, SUNK COSTS(e.g.PAST R&D) ARE NOT FACTORED INTO THE LCC COMPARISON. THREE BUYS OF 20 UNITS WILL BE MADE, ONE IN EACH OF THE NEXT THREE YEARS. THE SYSTEMS PURCHASED WILL BE OPERATED FOR THE REMAINDER OF THE TEN YEAR GAME.

THE B-52G MOD IS STILL IN THE R&D PHASE. THE COSTS USED FOR THIS ANALYSIS WILL BE THE LATEST ESTIMATES. INCLUDED IN THE ANALYSIS ARE THREE BUYS OF 20 UNITS, ONE IN EACH OF THE FIRST THREE YEARS THE SYSTEM IS AVAILABLE. IF YOU HAVE ALREADY BEGUN R&D, ONLY THE REMAINING COSTS OF DEVELOPMENT WILL BE CONSIDERED SINCE OUR CONCERN HERE IS FOR THE VALUE RECEIVED FROM CURRENT AND FUTURE AVAILABLE FUNDS. THE SYSTEMS PURCHASED WILL BE OPERATED FOR THE REMAINDER OF THE TEN YEAR GAME.

ANALYSIS YEAR 1
B-52G vs B-52G MOD

| YEAR | COST PER YEAR | | UTILS PER YEAR | |
|---|---|---|---|---|
| | B-52G | B-52G MOD | B-52G | B-52G MOD |
| 1 | 1200 | 3200 | 0000 | 0000 |
| 2 | 2000 | 4600 | 400 | 1600 |
| 3 | 2800 | 6200 | 800 | 3060 |
| 4 | 2400 | 4800 | 1200 | 4260 |
| 5 | 2400 | 4800 | 1200 | 4260 |
| 6 | 2400 | 4800 | 1200 | 4260 |
| 7 | 2400 | 4800 | 1200 | 4260 |
| 8 | 2400 | 4800 | 1200 | 4260 |
| 9 | 2400 | 4800 | 1200 | 4260 |
| 10 | 2400 | 4800 | 1200 | 4260 |
| TOTALS | 22800 | 47600 | 9600 | 34450 |

UTILITY PER DOLLAR FOR B-52G 0.421053
UTILITY PER DOLLAR FOR B-52G MOD 0.724370
DO YOU WISH TO ANALYZE ANY OTHER SYSTEMS? (y/n)n

Figure 12. Trade-off Analysis Report Display

ALTHOUGH IT IT NOT EASY TO RECOVER FROM ENTRY ERRORS
OR RECONSIDERATIONS, IT IS NOT IMPOSSIBLE. RECOVERY IS
ONLY POSSIBLE IF IT IS DONE IN THE SAME YEAR THAT THE
ORIGINAL ENTRY WAS MADE.
EACH ENTRY MODE CAN BE RESET INDIVIDUALLY (i.e. BUY,SCRAP,etc.)
AS DESCRIBED BELOW.

BUY
TO RESET THE QUANTITY OF A PARTICULAR SYSTEM YOU HAVE BOUGHT
YOU MUST FIRST FOLLOW ALL PROCEDURES AS IF YOU PLANNED TO BUY
THE SYSTEM AGAIN. HOWEVER, WHEN YOU ARE ASKED HOW MANY YOU
WANT YOU MUST ENTER THE NEGATIVE OF HOW MANY YOU PREVIOUSLY
ORDERED. THIS RESTORES THE FUNDS WHICH WERE DEDUCTED FROM
YOUR BUDGET. THEN IF YOU WISH TO ORDER A DIFFERENT NUMBER,
SIMPLY REORDER IN THE NORMAL MANNER. NOTE!!!IF YOU DO NOT
WANT TO BUY ANY OF THE SYSTEMS AFTER YOU HAVE RESTORED YOUR
BUDGET YOU MUST STILL FOLLOW THE ORDER PROCEDURE AND ORDER
A QUANTITY OF ZERO.

MODIFY(UPGRADE)
THE SAME PROCEDURE MUST BE FOLLOWED TO RESET THE NUMBER OF
SYSTEMS TO BE UPGRADED USING THE MODIFY COMMAND. CARE SHOULD
BE TAKEN, HOWEVER, TO ALSO ENTER THE NEGATIVE OF THE NUMBER
OF SYSTEMS YOU ASKED TO HAVE CONTINUED AT THEIR OLD COST AND
UTILITY.

SCRAP
TO RESET THE NUMBER OF SYSTEMS YOU WANT TO SCRAP, SIMPLY
FOLLOW THE PROCEDURE TO SCRAP A SYSTEM. WHEN IT ASKS YOU
HOW MANY YOU WANT SCRAPPED ENTER THE NEW VALUE YOU WANT TO
USE. IF YOU HAVE DECIDED NOT TO SCRAP ANY, ENTER ZERO.

INTELLIGENCE
TO RESET AN ENTRY IN THE INTELLIGENCE FUNCTION YOU MAY CHOOSE
THAT OPTION FROM THE MENU IN THAT SUBROUTINE.

Figure 13. Recovery from Entry Error Display

12. Submit Completed Planning and Budgeting

When the team has completed all of its actions and requests for the year, the final selection on the menu is entered. It should be noted that each team is given a specific time to use for current and long-range planning. It is strongly recommended that the teams take full advantage of their allotted time to carefully prepare their plans. There is no advantage to an early submission of the yearly plan. The computer will simply make the team wait for the end of the period anyway. It should also be emphasized that a team is penalized for a late submission. Significant dollar reductions from the next year's budget are assessed for each minute the plan is delinquent.

Upon selection of the year-end option from the menu, the teams are asked to confirm that they, indeed, are done for the year. At this point they have the option of returning to the main menu for subsequent adjustments.

When a confirmed entry is made, the year-end module performs several processes in closing out accounting for the year. Figure 14 shows the first report which provides a summary of utils provided by the weapon systems and dollars spent for the year. This report is immediately displayed at the team's terminal.

Next, the teams are advised of a break period to allow them to relax and collect themselves for the next planning cycle. During the break, the umpire program is called up

```
IF YOU ARE CERTAIN THAT YOU HAVE MADE ALL THE ENTRIES
THAT YOU WANT FOR THIS YEAR, CONFIRM WITH A 'YES'.
OTHERWISE, 'NO' WILL RETURN YOU TC THE MENU. (y/n)
y

HERE IS A BRIEF SUMMARY OF YEAR 1 .

YOUR TOTAL OFFENSIVE A UTILS WERE 750.
YOUR TOTAL OFFENSIVE B UTILS WERE 800.
YOUR TOTAL DEFENSIVE A UTILS WERE 1000.
YOUR TOTAL DEFENSIVE B UTILS WERE 1000.

THE TOTAL OPERATIONS COSTS FOR YOUR SYSTEMS WERE 5600.

YOUR TOTAL EXPENDITURES FOR THIS YEAR WERE 6800.

THIS IS THE END OF YEAR 1. FOLLOWING A
FIFTEEN MINUTE BREAK, YOU WILL BE GIVEN 45 MINUTES
TO PREPARE YOUR PLAN FOR YEAR 2. IF YOU REQUESTED
INTELLIGENCE DATA YOU WILL RECEIVE IT AT THE
BEGINNING OF NEXT YEAR'S PLAY.

THIS YEAR WAS A DRAW
```

Figure 14. Year End Display

and the data are fed to a central point for recording and determination of the winner for the year. The umpire program uses a random number routine to determine whether a war has occurred during the year. If a war has broken out, messages to that effect are written into files and later retrieved by both sides. The activities of the umpire program are all transparent to the players of the game. Finally, the year-end routine updates the database for the coming year, and passes intelligence data back and forth between the red and blue sides.

When the program becomes active again, an introductory report for the next year will appear on the terminal screen (see figure 15). At this time, the intelligence reports which were requested during the previous year are also available simply by pressing the carriage return key on the terminal keyboard. A sample intelligence report is shown in figure 16. Following review of the intelligence reports, the teams receive the original menu and repeat the process for the next year.

D. WAR

Throughout the game there is a threat that war could break out in any year. The intelligence sources for each side will keep the teams informed of the best estimate of probability of an outbreak of hostilities for each year. This information is provided at the beginning of each year

```
THIS IS YEAR 2 OF THE GAME.

YOUR BUDGET FOR THIS YEAR IS $8000.
        BUDGET ESTIMATE FOR NEXT YEAR IS $8500.
        BUDGET ESTIMATE FOR THE FOLLOWING YEAR IS $9500.

THE PROBABILITY OF WAR FOR THIS YEAR IS 0.15
        INTELLIGENCE ESTIMATES THE PROBABILITY OF
        WAR FOR NEXT YEAR TO BE 0.25

IF YOU ORDERED INTELLIGENCE REPORTS, THEY WILL
        APPEAR NEXT ON THE SCREEN.


FOR MORE DETAIL ON THIS YEAR'S FORCE STATUS AND
        PROGRAMS SELECT THE DESIRED ENTRY FROM THE
        FOLLOWING MENU.


PRESS RETURN TO CONTINUE.
```

Figure 15. Sample Year Introductory Report Display

BLUE OFFENSIVE FORCE STATUS IS:

| | |
|---|---|
| B-52G | 040-049 |
| B-52G MOD | 000-009 |
| FB-111H | 000-009 |
| B-1 | 000-009 |
| MM-III | 020-029 |
| MM-III MOD | 000-009 |
| ALCM | 000-009 |
| MX | 000-009 |
| FOB | 000-009 |

THIS INTEL REPORT LISTS ALL OFFENSIVE PROGRAMS FOR WHICH R&D FUNDS HAVE BEEN SPENT. THIS YEAR.

B-52G MOD
FB-111H

NOTE: YOU WILL NOT SEE THESE REPORTS AGAIN SO YOU WOULD BE WELL ADVISED TO RECORD THE DATA YOU NEED.

Figure 16. Sample Intelligence Report Display

and need not be requested or funded. It is also available in the BUDGET module at any time during the year. Wars will not last more than one year at a time.

If a war occurs, the cost in resources for each team will be reported to them and a winner will be declared. The format for this report is shown in figure 17. The overall winner of the game, however, will be the team with the most net offensive utils for the entire eight-year game period.

E. PENALTIES

If a war occurs both teams will have $400 cut from the budget for next year. In addition, the loser's budget will be cut by an amount equal to twice the difference between the two teams net offensive util count calculated for the year the war occurred.

If a team allocates more funds than it has in its budget during a given year, the budget in the following year will be reduced by twice the amount of the deficit. In addition, as soon as a team's expenditures (allocations plus operating costs) exceed the budget, the program will take it automatically to the year-end module. The team has the option of returning to the main menu. However, with each additional expenditure it will once again be sent to the year-end module. Advisories will be provided to explain what is happening but the awkwardness of this condition will presumably prevent significant overexpenditures.

```
W          W  AAAA  RRRRR
W     W    W A    A R   R
W   W W    W A    A RRRRR
W  W   W   W AAAAAA RRR
 W W    W W  A    A R  R
  W      W   A    A R   R
```

HAS BROKEN OUT THIS YEAR!!

IT WAS A SHORT-LIVED CONFLICT AND YOU FOUGHT TO A STANDSTILL..
WITH YOUR OPPONENT. THE COST OF THIS ENCOUNTER WILL BE A $400
REDUCTION IN YOUR BUDGET FOR NEXT YEAR.**********
**********
**********
**********

<CR> to continue

Figure 17. Sample War Report Display

If a budget is submitted late (that is, if the end-of-year menu selection is not made in the allotted time) the budget for the next year will be reduced by $50 per minute for the first five minutes, and $100 per minute for each minute over five minutes.

If a team does not spend its entire budget for the current year, the remainder of the funds are lost. There is no carryover to the following year.

F. WEAPON SYSTEM EFFECTIVENESS

The total utility of a given weapon system for one year is not simply the number of utils times the number of systems. The Law of Diminishing Marginal Returns postulates that stockpiling one particular weapon system which has a high utility does not insure success, nor does it support a balanced force concept. Therefore, the final util count for a system in a particular year may be reduced by some amount. The scheme for this util discount is shown in figure 18. For example if a team has 100 weapon systems, each with a utility of 55, the team will receive a total utility for that system of 4400 + 300 = 4700 rather than 5500.

G. DOCUMENTATION

Teams will be provided with blank budget allocation forms to assist them in planning where resources will be spent for the year. These forms need not be submitted at any time. They are merely tools to make the organization of

FORCE UNIT UTIL DISCOUNT

Total utils of any individual force unit are discounted as follows:

| TOTAL ORIGINAL UTILS | FINAL NET UTIL VALUES |
|---|---|
| 1 - 2000 | original total |
| 2001 - 3000 | 2000 plus 90% of amount over 2000 |
| 3001 - 4000 | 2900 plus 80% of amount over 3000 |
| 4001 - 5000 | 3700 plus 70% of amount over 4000 |
| 5001 - 6000 | 4400 plus 60% of amount over 5000 |
| 6001 - 7000 | 5000 plus 50% of amount over 6000 |
| 7001 - 8000 | 5500 plus 40% of amount over 7000 |
| 8001 - 9000 | 5900 plus 30% of amount over 8000 |
| 9001 - 10000 | 6200 plus 20% of amount over 9000 |
| 10001 - 11000 | 6400 plus 10% of amount over 10000 |
| over 11000 | 6500 maximum net utils |

Figure 18. Utility Discount Scheme

the teams easier. A sample budget allocation form is shown in figure 19.

Each year both teams receive an actual budget for the current year plus estimated budgets for the next two years. These estimated budgets may change, but they can assist in making some long-range plans. The Multiple Year Defense Program(MYDP) is patterned after the Five Year Defense Program and is provided to help the teams formulate long-term force plans. The MYDP is merely a worksheet and will not be submitted. A sample MYDP is shown in figure 20.

E. UMPIRE

The umpire has only one responsibility in the play of this version of TEMPO. He must signal the start of play when both teams have had the chance to review initial game data.

The umpire can also observe game status in terms of each team's utility totals for each year by logging into 'umpire' with password 'umpire' and examining the file "ump" (type 'aa ump" once in the directory). This is a cumulative file which is updated at the end of each year. Note that the file must be recalled at the end of each year to receive the most current data. A file being displayed will not be automatically updated as it is being viewed.

Year___ Budget_____

OPERATIONS

| SYSTEM | BEGIN YR INVEN | # TO SCRAP | # TO MOD (NOP) | NUMBER TO OPERATE | OPER COST PER UNIT | OPER COST TOTAL | TOTAL UTILS |
|---|---|---|---|---|---|---|---|
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | Total Operating Cost | | |

RESEARCH AND DEVELOPMENT

| System | R&D Cost |
|---|---|
| | |
| | |
| | |
| | |
| | |
| Total R&D Cost | |

ACQUISITION

| System | # to Buy or Mod | Unit Cost | Total Cost |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | Total Acquisition Cost | |

INTELLIGENCE

| Type | Intel Cost | Counter |
|---|---|---|
| Off Force | $100 | $50 |
| Def Force | 100 | 50 |
| Off R&D | 100 | 50 |
| Def R&D | 100 | 50 |
| Subtotals | | |
| Total Intel Cost $ | | |

ANALYSIS

($25 per analysis)

| System | vs | System |
|---|---|---|
| | | |
| | | |
| | | |
| | | |
| Total Analysis Cost_____ | | |

Figure 19. Sample Budget Allocation Form

| SYSTEM | Year__ | | Year__ | | Year__ | | Year__ | | Year__ | | Year__ | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | # | $ | # | $ | # | $ | # | $ | # | $ | # | $ |
| OPS | | | | | | | | | | | | |
| ACQ | | | | | | | | | | | | |
| R&D | | | | | | | | | | | | |
| MOD | | | | | | | | | | | | |
| OPS | | | | | | | | | | | | |
| ACQ | | | | | | | | | | | | |
| R&D | | | | | | | | | | | | |
| MOD | | | | | | | | | | | | |
| OPS | | | | | | | | | | | | |
| ACQ | | | | | | | | | | | | |
| R&D | | | | | | | | | | | | |
| MOD | | | | | | | | | | | | |
| OPS | | | | | | | | | | | | |
| ACQ | | | | | | | | | | | | |
| R&D | | | | | | | | | | | | |
| MOD | | | | | | | | | | | | |
| OPS | | | | | | | | | | | | |
| ACQ | | | | | | | | | | | | |
| R&D | | | | | | | | | | | | |
| MOD | | | | | | | | | | | | |
| Totals Est Budget | | | | | | | | | | | | |
| Difference | | | | | | | | | | | | |

Figure 20. Sample Multiple Year Defense Program Form

## IV. PHILOSOPHY OF THE GAME

### A. TEAM ORGANIZATION

There are no limits on the number of players who may play this version of TEMPO. One player can familiarize himself with the program through the demonstration version. Two players can learn from the exercise in either the short or long version. However, the real value of the game is gained when it is played by two teams each with several players. With a staff organization on both sides, the players can begin to appreciate the challenges of orchestrating an annual budget. The operational commanders must cooperate with the budgeteers and the schedule monitors must ensure that everyone understands the criticality of time.

There is no optimal organizational structure for all teams. It is recommended that one of the players on each team assume the role of team leader, or commander.

Some steps which might be helpful in establishing a team framework are analyzing the problem, setting an overall goal, developing a plan, and organizing.

#### 1. Analyzing

Before the teams enter the lab they should carefully review the User's Manual as presented in Appendix A. This gives enough background to provide them a good idea of what

it is they are trying to accomplish. It describes the constraints they must work under and the opportunities they will have. A group discussion of the various interpretations of this manual should give the group a good fundamental idea of the problem.

2. Goal Setting

It may be helpful for the teams to establish a tangible quantitative goal at the cutset of the game. "To win the game" is an admirable goal but without a more definitive measure (or guess) of what it will take to do so the team has achieved no benefit. There is no harm in adjusting goals as the game progresses but the teams should try to establish some desired level of preparedness for a point in the future. Failure to do so can sometimes result in aimless wandering in search of an appropriate force.

3. Planning

Once goals are set the team should begin to plan how they will attempt to achieve those goals. If the approach chosen is to make maximum use of intelligence sources to plan R&D programs, the team should fortify its planning for such. If the team chooses to amass forces already available it should establish a plan for doing so. These plans are relatively easy to initiate simply by concentrating the staff resources at the point where the strongest thrust is desired. The teams should try to anticipate potential problem areas and have contingencies to back them up. It

may also be helpful at this stage to establish procedures for a consistent approach to planning and analyzing throughout the game. Analyses must be reproducible and R&D procedures should be consistent throughout.

4. Organizing

The Squadron Officer's School curriculum proposes a five-step process of organizing a TEMPO team once the initial planning is complete. First, all objectives should be listed based on goals and plans. Second, the objectives should be grouped into functional areas. Third, a ranking of these objectives is established based on the relative importance of the functions. Next, a functional organization chart may be sketched to see how the pieces interact. And finally, available staff members should be assigned to the functional areas. Questions may arise regarding decision-making procedures, span of control, specialists in certain areas, and functional group interfaces. When the questions have all been answered, the team is ready to begin the game.

B. TIME LAG

Most observers of the Department of Defense system acquisition process agree that ten years is a good estimate of the time it takes a weapon system to develop from an idea to a combat system in the field. This time lag is one of the parallels of reality included in TEMPO. Although the

development time will more likely be two or three years rather than ten, the teams must use some foresight to anticipate the benefits to be gained by developing new systems. Figure 21 shows a utility comparison, similar to those available in the ANALYZE module of the game, which illustrates trade-offs which must be made.

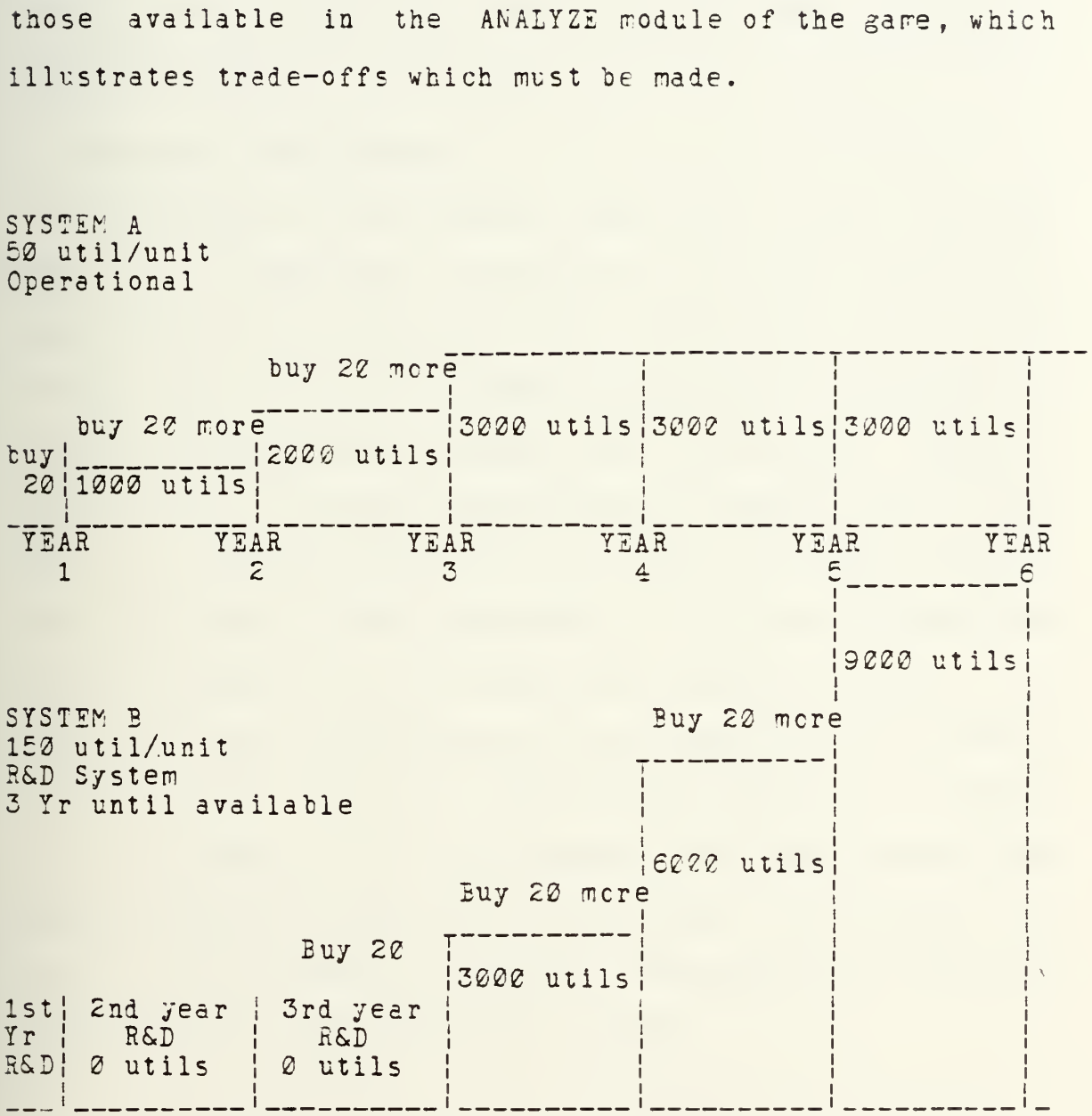


Figure 21. Utility Comparison

The teams must evaluate the probabilities of war in the early years as compared to the benefits to be gained in the long run before making the decision to buy or R&D. However, they should not lose sight of the fact that net offensive utility over the eight year period of the game determines the winning team.

C. ANALYZING THE OPPONENT

TEMPO uses a very simple algorithm in its analysis module for determining life-cycle utility per dollar of a candidate system. The teams should observe that this same formula can be used to analyze and bound the opponent's capabilities using data obtained from intelligence reports. Specifically, if a team learns that its opponents have 10-19 units of a system with 25 utils per unit and a $40 operating cost per unit, it can deduce that it must have between 250 and 475 defensive utils to offset the opponent's force. It also knows that the opponent must use $400-760 of his budget to operate that force. By anticipating what new systems will be coming into the inventory from R&D programs and making insightful guesses as to how many the opponent is likely to buy based on his budget constraints, an effective counterforce can be planned and fielded.

D. HINTS

There are a few insights which could prove helpful during the play of TEMPO. Teams will undoubtedly discover many more as they gain experience with the game.

1. The most useful piece of advice is to carefully read the User's Manual before beginning to play the game. The constraints imposed by TEMPO do not allow time for familiarization during the planning periods.
2. The teams may want to investigate linking a hardcopy miniterm to the war gaming terminal during play so that a permanent backup copy of reports is available for staff planning.
3. If a team overspends its budget during a given year there is only one way to reduce current year expenditures. In the MODIFY module the team is asked if it wants to take current systems off the line while they are being modified. If the team so chooses, the operating costs for the current year are eliminated. In this way a limited amount of operating costs for the year can be reduced. This is a subtlety which the instructor may wish to let the teams discover for themselves.
4. Instructors should also beware of students who have prior experience with the game. These students can be helpful in explaining details to their teammates.

However, if one team has prior experience while the other does not, an undesirable imbalance could result.

## V. STRUCTURE OF THE GAME

### A. PROGRAM ARCHITECTURE

The computer program which implements TEMPO in the C3 Laboratory was written in C Language on the UNIX Operating System. The computer is a PDP Model 11/70. Ann Arbor terminals were used for development and are recommended for play of the game. A minimum of three terminals are required for the game. With minor modifications for screen size, the program is transportable to any other comparable system.

TEMPO is composed of three primary elements. These are the Red Force program, the Blue Force program, and the Umpire program. The top level computer program architecture is depicted in figure 22. Note that the BLUSIDE program actually calls the UMPIRE program while the REDSIDE program merely provides data to the UMPIRE.

Figure 22. Top Level Architecture

The two primary programs, REDSIDE and BLUSIDE, are accessed by logging into the respective war-gaming stations with login names 'tempored' and 'tempoblu' respectively. Passwords are 'tempor' and 'tempob'. Once logged in, the

players initiate the program by typing 'begin'. The umpire reports are accessed by logging into the directory 'umpire' with password 'umpire'. There is no user interaction with the umpire program other than to read its resident files. The umpire program is called and executed by BLUSIDE at the end of each game year. REDSIDE furnishes its data to UMPIRE by writing it into a file which UMPIRE later reads. A more detailed computer flowchart is provided in figure 23.

The first nine subroutines below each major program in figure 23 are the menu selections each side is afforded during the yearly planning cycles. These were described functionally in Chapter 3. The code and program descriptions are provided in the Computer Program section at the end of this thesis. Detailed descriptions of the subunits of the DONE subroutine are also contained in the Computer Program section. An important point for discussion here, however, is the way in which data is passed among the three main programs in the game.

One of the options available to the players is the purchase of intelligence or counterintelligence data regarding the opponent's status. In order to do this each side writes its force structure and R&D activity for the year into files (BLUOFF, BLUDEF, BLURDO, BLURDD, CNTRINT, etc.). as shown in figure 23. These files are made available to the other side through selection in the INTREP subroutine. The reason for five separate files for each

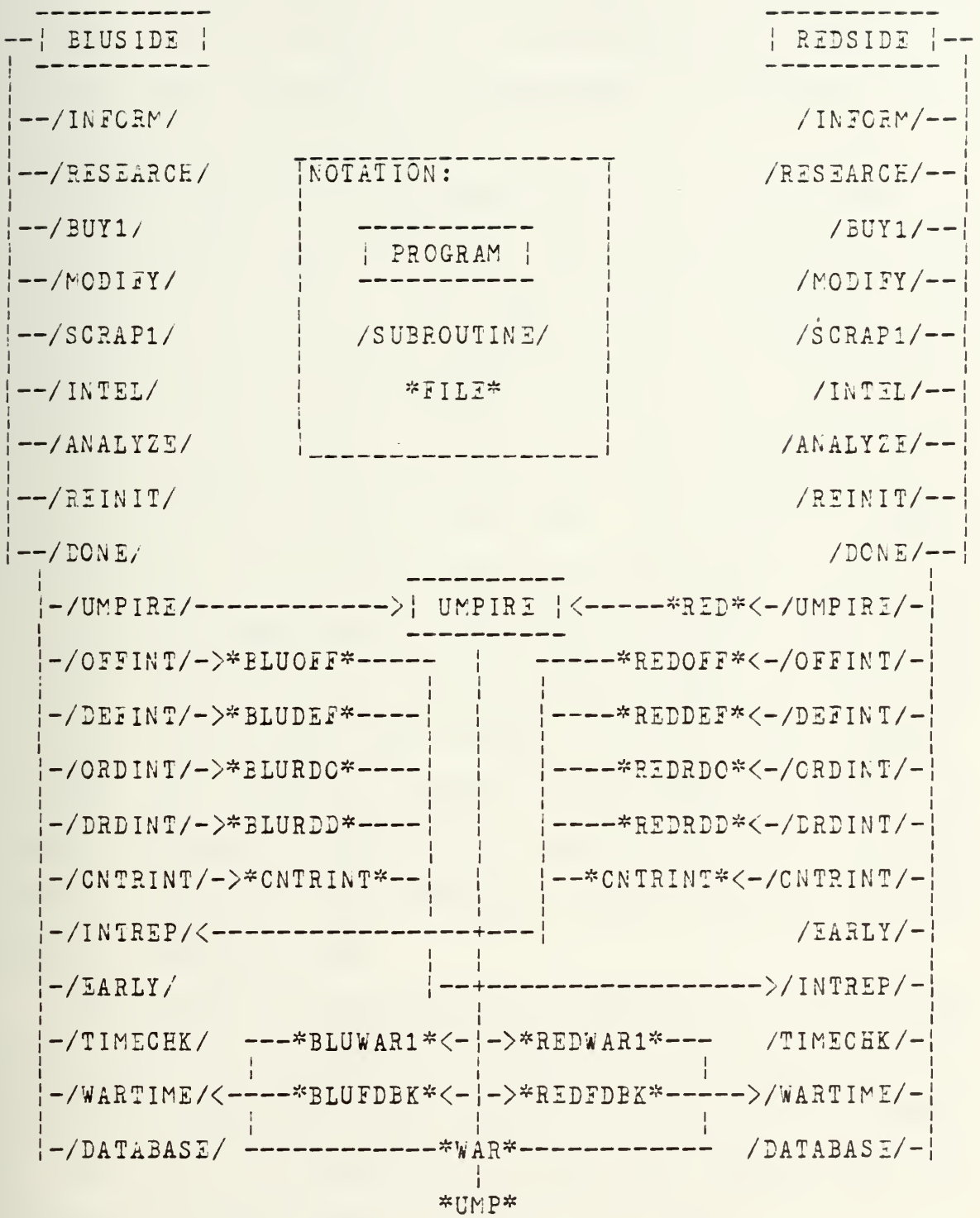


Figure 23. Detailed Program Functional Diagram

side is that the teams are charged budget dollars for access to each type of information. If all information were put in the same file the denial of nonrequested information would be considerably more difficult.

Another aspect of the game which bears mention here is the possibility that a war will occur during any year. The probability of war is an input value. However, the occurrence of a war is determined by a random number generated in the UMPIRE program. If the random number falls in the designated probability window, war is declared and a flag is set in the umpire file WAR. The UMPIRE program automatically selects and sends the appropriate win or loss messages and budget reductions to the two teams. Both team programs read the WAR flag at year's end. If a war has occurred they read the results in the files BLUWAR1, BLUFDBK, and REDWAR1, REDFDBK respectively. BLUWAR1 and REDWAR1 describe the outcome of the war and BLUFDBK and REDFDBK provide the necessary budget penalties to be deducted from each side.

The datafile UMP generated by UMPIRE is a cumulative accounting of offensive and defensive status for each team by year for the entire game. This is available for review by the umpire by logging into 'umpire' and reading the file through the editor (type "ea ump").

The DATABASE subroutine is read at the beginning of each year to update weapon system costs and characteristics as they change throughout the game.

## B. LESSONS LEARNED

There were some significant problems encountered during the coding of TEMPO and some lessons learned which may help future C Language programmers on the UNIX system. The most pressing problem for a considerable part of this effort was a perceived program size constraint. Much of the structure of the present programs was dictated by an apparent lack of program space. The perilous error message output by the system when this problem is encountered reads "ld: Bad Format". Late in the programming phase it was discovered that by using the compiler option "-i", this problem can be virtually eliminated. To use this option the programmer simply must partition his program into two or more smaller segments, and then, when compiling, type "cc -i segment1.c segment2.c etc.c" instead of the usual "cc entire.c".

Another difficult problem was timing the data transfers between subprograms and subroutines of the opponents and of the umpire. This was solved by writing the current data into files early in the year-end process and reading from the appropriate files later. To ensure that both programs begin at the same time, the EARLY subroutine was inserted. This stops the processing of the teams in the event that

they finish their entries before the allotted time is used up. It also keeps the entire game on a strict time schedule so that small offsets early in the game do not grow into large offsets over the eight year game period. The possibility that a team uses more than its allotted time was addressed by a statement in the umpire program which inhibits further processing until the year tag on each team's data is equivalent. The subroutine TIMECHK also checks to see if a team was delinquent in its year-end submittal. Furthermore, since the last four years of the game are program years and all data for the four years is submitted in one planning period, these data are sent to separate files for each year to ensure that the proper comparisons are made.

Most of the other problems encountered were solvable through a mixture of perseverence and a return to good structured programming practices.

## VI. RECOMMENDATIONS

Although the TEMPO game is fully functional at this time, there are several improvements which could be incorporated to 'dress it up'. One goal which was not fulfilled was the incorporation of computer graphics into the program. With the extensive color graphics capabilities in the C3 Laboratory, the potential for graphical representation of many of the team reports is very exciting. Force levels, intelligence reports, and war declarations are only a few of the possible areas for graphical presentations.

An area which has a great potential for improving the reality of the game is the determination of weapon system utility. Currently, this is a database input selected arbitrarily by the game designer. In the present implementation, the total utility of a force of particular weapon systems is a function only of the number of systems and the defense of the opponent. In reality, utility of a weapon system is also a function of the number of corresponding offensive systems the opponent has in his inventory. An example of this is the seeming decline in utility of U.S. ICBM forces due to increasing numbers of Soviet ICBM forces. Utility is also a function of the number of delivery systems available for the weapon system

under consideration. Or, if the weapon system under consideration is a delivery system, its utility is a function of the number of weapons available. In addition, utility could be considered a function of the number of other types of systems the team has in inventory. For example, a force composed entirely of bombers probably does not provide as much utility as a balanced force of missiles and bombers. The development of a utility module that would consider all of these factors in determining utilities would be a challenging exercise which would lend greater realism to the TEMPO game.

Another area which could be improved upon is the determination of the probability of war. Presently, this is also a database input chosen to give the teams a year or two to get a feel for the game, and then be fairly well assured that a war will occur at least once during the game. To be more realistic a set of thresholds for increasing war probabilities could be developed. For example, if a team has an extremely low level of defensive forces and the other team finds out through an intelligence report, it would be reasonable to expect that the probability of war would increase. If a team determines that it could bankrupt its opponent through an overwhelming battle victory, the likelihood of an attack should increase. If a team displayed an immense offensive buildup in one year or in successive years, that could signal an increase in the

likelihood of war. These thresholds would be relatively easy to determine and fairly straightforward to implement in the code.

# APPENDIX A

## TEMPO USER'S MANUAL

### I. INTRODUCTION

The TEMPO Military Planning Game was originally developed in the early 1960's by the Technical Military Planning Organization (TEMPO) of General Electric Company in Santa Barbara, California. Since that time it has been used by the Defense Resources Management Course at the Naval Postgraduate School; the U.S.Army Management School, Ft.Belvoir, Virginia; the Air War College, Air Command and Staff College, and Squadron Officers' School all at Maxwell AFB, Alabama. Additionally, many variations of the game have been developed and used throughout industry and the government.

Play of the game may be approached from several different perspectives ranging from simple diversion to studies of interpersonal behavior. Its primary purpose, however, is to train military leaders in the skills of force planning and resource management under the realistic constraints of time, uncertainty, and a limited budget. These skills are particularly important to officers entering the world of Command, Control, and Communications(C3) since the timely fielding of C3 systems is crucial to the strategic and tactical security of our military forces.

The manipulation of the strategic forces in this game has been greatly simplified to focus the attentions of the players on the force planning and resource allocation aspects of the situation. These aspects include the selection of alternative weapon systems, the maintenance of proper force mixes, the total life cycle costs of the systems, cost effectiveness considerations, long range planning, and the elements of risk and uncertainty.

The game uses actual weapon systems as the objects of this planning. No attempt has been made to exhaustively analyze the effective utility of a specific red force system against its corresponding blue force system. Opposing systems were chosen to be "roughly" equivalent in utility, and the names are included solely to stimulate player interest in the game.

There are few features of this version of TEMPO which are unique. It is based primarily on the game as played at Air University in Montgomery, Alabama. Its uniqueness lies in its implementation on the PDP 11/70 computer resident in the C3 Laboratory at the Naval Postgraduate School. This provides the players with the capability to interact in real-time with the program through the war-gaming stations in the laboratory.

II. GENERAL INSTRUCTIONS

TEMPO is played with two teams, a red force and a blue force, and an umpire. Both teams start with identical force structures, research and development options, and budgets. That is to say, the costs, utilities, and research and development potentials for corresponding systems on each side are the same even though the systems have different names.

The budget, which varies from year to year, may be spent in any of six ways: (1) operation of forces, (2) acquisition of additional forces, (3) research and development of new systems, (4) modifications to existing weapon systems, (5) intelligence or counterintelligence, and (6) trade-off analyses between systems.

The game spans eight years with the first four years being operational years and the last four years being programmed years. All transactions must be completed within the allotted time, or budget penalties for the ensuing year will be incurred.

The categories of systems which each team has to work with are limited to offensive strategic aircraft, offensive strategic missiles, surface-to-air missiles (aircraft defense), and defensive anti-ballistic missile (ABM) missiles. Several types of weapon systems are available in each category through the course of the game. System capability is quantified in terms of a standard measure of

effectiveness called the "util". Each weapon system is worth a specific number of utils while it is in the active force. The author recognizes that one of the key factors in determining proper force structure is the effectiveness of one's own forces versus the opposition they are most likely to encounter. However, for the purposes of this exercise, utility will be treated as a given, thus allowing concentration on the budgetary aspects of the situation.

The objective of the game is for each team to operate its offensive forces to maximize its net offensive utils while operating a defensive force to minimize the opponent's net offensive utils. The limited budget which each team must work with increases the challenge of this objective. The teams must decide if it is worthwhile to allocate scarce budget dollars to research and development for improvement of the force efficiency. Dollars spent for programs other than strictly procurement of existing systems may result in less than optimum current force structures in order to improve future balance. In addition, teams must decide whether or not to invest in intelligence about their opponent. An analysis option for making cost-benefit tradeoffs is also available to each team.

There are three versions of TEMPO available in the C3 laboratory. The normal version requires four hours to complete the game. This mode allows for extensive planning periods which are desirable for larger teams to derive the

full benefit of cooperative staffing in the various areas of concern. The short version requires two hours for completion and has much shorter planning periods. This mode allows the teams to get an exposure to the features of the game under an extreme time constraint. The third version is a demonstration program to allow a single user to examine the features of the game. There are no time constraints involved in the demonstration program.

## III. DETAILED RULES

### 1. OBJECTIVE

The objective of the game is for each team to operate its offensive forces to maximize its net offensive utils while operating a defensive force to minimize the opponent's net offensive utils. The formula for determining a team's net offensive utils is:

(total offensive aircraft utils minus opponent's total anti-aircraft utils)

plus

(total offensive missile utils minus opponent's total anti-missile utils,

The net offensive utility resulting from this calculation is compared to the opponent's net offensive utility to determine who wins each year. The teams will be notified at

the end of each year whether they won or lost that particular year. The overall winner of the game is the team which has the greater cumulative total of net offensive utils for the eight-year period.

The players should bear in mind that utils are only received for systems which are in the active force. No credit is received for over-defending. That is, it does no good to have more defensive utils in a particular weapon system category than the opponent has offensive utils. And finally, anti-aircraft systems are only effective against aircraft, and anti-ballistic missiles are only effective against offensive missiles.

2. STARTING THE GAME

To begin the game, the teams should login to the computer at their respective stations, i.e. the Kremlin at one end of the laboratory and the Pentagon at the other end. The computer directories are 'tempored' and 'tempoblu' with passwords 'tempor' and 'tempob' respectively. The demonstration version of TEMPO is accessed by logging into 'tempdemo' with password 'tempod'. Upon login the teams will be asked to indicate whether they want to play the short or long version of TEMPO. When they have provided their response the teams will receive some introductory remarks including the current force structure and budget for the first year, estimated budgets for the next two years and the probability of war. Proposed research and development

programs for the first year are also displayed. This information is available at any time during the game so there is no need to record it.

When the teams have had sufficient time to review the initial data the umpire will signal the beginning of the game. To initiate play the teams press the carriage return keys at their respective stations and the game commences.

IMPORTANT! Since timing is critical to this implementation of TEMPO it is imperative that the teams wait for the umpire's signal to begin.

The next display to appear on the screens is a menu of options available for the teams to choose from. These menu items will be described in detail in the following paragraphs. NOTE: The menu items may be reselected as often as desired during each year.

3. CURRENT FORCE STATUS AND DATA

This segment of the program provides a summary of all systems in the current operating inventory. The table includes number of systems in operation, acquisition cost, operating cost and utility of each system. The purchase limit per year is also listed for the systems.

4. RESEARCH AND DEVELOPMENT DATA

The first display which appears when this option is selected shows which research and development (R&D) programs

are available for that particular year. The teams have only one year to begin work on a particular program. If they do not select the option they will not get the chance the next year. There is one exception to this rule. The R&D required prior to a system modification can be done in any of the first three years.

New research programs are offered only in years one, two, and three of the game. Programs begun in these years may be continued to completion but no new programs will be started beyond year three.

The teams will be asked by the computer if they wish to see a complete R&D report for the year. This report provides estimates of acquisition cost, operation cost, utility and R&D costs for each available program. The teams will be told how long the development will take and what the expected R&D costs for the future years of the program will be. The current year R&D cost is actual, however, the second and third year costs (if the program takes that long) are only estimates and may increase or decrease depending on how the program progresses.

Finally, in this module, the teams are asked if they wish to spend funds for any of the programs. If the answer is yes a third display appears with a menu of candidate R&D programs. The actual and projected R&D costs are again shown along with a status report of ongoing programs. If the team has begun a project in a previous year they are

reminded to continue the funding or else jeapordize the delivery schedule. The teams do have the option to cease funding of a program after the first year of R&D has been completed and then resume the development in a later year.

Additional R&D information on a system will be available only if the team initially chooses to pursue the program.

5. CURRENT BUDGET DATA

This module provides each team with the total budget figure for the year, the total cost of operating the force for the year, and a running total of dollars spent up to that point in the year. Budget estimates for the next two years are also provided. Actual allocations for future years may vary but the estimates can provide a basis for planning. The probabilities of war, as currently estimated, are provided in this module.

The teams are advised to keep a close account of financial status since overspending the budget results in a penalty assessed against the next year's budget.

6. BUY ADDITIONAL SYSTEMS

Selection of this option provides each team with a list of all systems which are available for purchase. Included in the list are the acquisition costs, utility, and purchase limits for each system. The players simply choose which system they want and the computer will ask how many the team wants to buy. When that input is provided the players receive acknowledgement of the sale.

Systems bought in one year will not be operated until the next year. Consequently, no operation costs will be incurred and no utility will be gained until the following year.

A new system is available for purchase as soon as all the R&D has been selected and paid for. A team may acquire units of a new system during the last year of R&D (provided the R&D has been funded), or at any time thereafter. A team cannot lose its option to buy units of a system once it is in the inventory.

7. MODIFY SYSTEMS IN CURRENT INVENTORY

At certain points in the game the teams may be offered the opportunity to modify or upgrade weapon systems they have in their inventories. These modifications are not possible without developing the technology which goes into them. Therefore, the R&D for the modification must be funded before the modification can take place. This funding is provided through the R&D module described above.

When the modify option is selected the players will be offered candidate systems for upgrade. There are two ways a system can be modified. It can be withdrawn from operations for the year, thus incurring no operation costs and providing no utility. Or it can be operated at the unmodified levels until the next year when the modification is complete and it automatically assumes the upgraded levels. The teams must choose how many units they want to

modify and how they want to distribute the way they are modified. The computer will provide the necessary advisories.

To purchase upgraded systems directly, that is, other than upgrading systems already in the inventory, it is still necessary to pay the modification R&D cost (if it has not been paid). The new units can be purchased at a price equal to the procurement cost of the unmodified system plus the modification cost. This option will automatically be made available to the teams through the BUY module when the modification R&D is completed.

Modifications may be made at any rate not exceeding the purchase limit of the original system. R&D for the modifications may be performed at any time during the first three years of the game.

8. SCRAP SYSTEMS IN CURRENT INVENTORY

It may be necessary at certain junctures of the game to get rid of older, less-efficient systems in order to free up their operations costs for other purposes. This can be done by selecting the SCRAP menu choice. A complete list of systems in the current inventory is displayed and the team is asked which system they would like to scrap. They are then asked how many units they would like to take out of operation and the computer acknowledges the entry. Force units scrapped in one year will remain in the inventory until the following year.

9. BUY INTELLIGENCE DATA OR COUNTERINTELLIGENCE

Selection of this module provides the team with a menu of ten options for intelligence gathering. For $100 each the team can get a report of the opponent's:

a) current force structure of offensive forces, or

b) current force structure of defensive forces, or

c) current R&D programs for offensive systems, or

d) current R&D programs for defensive systems.

Intelligence reports on force structure do not provide an exact count of enemy systems. Rather, they present a ten-unit range for each system, e.g."SYSTEM-A has 20-29 units" is the report if the opponent has 22 units.

R&D intelligence reports list all programs for which R&D funds were expended that year by the enemy.

Either team may purchase counterintelligence in any of the four areas described above for $50 per category. Counterintelligence reports inform the team if the enemy has purchased intelligence in that category during the year.

The intelligence menu provides the option to cancel all orders for intelligence and start over again for the year.

If an intelligence report is requested it will be presented automatically at the beginning of the following year's planning period.

Intelligence reports cannot be obtained in years five through eight.

10. PERFORM TRADE-OFF ANALYSES

This module may be selected any time during years one through four to assist in making decisions as to which systems to buy. The analysis option provides a complete list of systems which are, or could be, available at that time for procurement or R&D. The team is asked to select two systems to be compared and an immediate analysis is presented. The report includes year-by-year life-cycle-cost breakouts for each system in the trade-off as well as yearly utility figures. Totals for both of these parameters are tabulated and used to compute a util/dollar cost effectiveness factor. This value is simply the total life cycle utility divided by the total life-cycle cost.

The teams will not be prevented from performing trade-offs using systems they may have chosen not to R&D. This kind of analysis is not very useful, however, since the preferred option may not be possible to obtain.

The cost of analysis is $25 for each trade-off performed.

11. RECOVERY FROM ENTRY ERRORS

This module is merely a tutorial to explain how to change entries the teams may have made and then later reconsidered. It is not possible to recover from every entry error, particularly in the R&D module. A team cannot

un-start development of a new technological program. Most input errors can be corrected, however.

12. SUBMIT COMPLETED PLANNING AND BUDGETING

When the team has completed all of its actions and requests for the year, the final selection on the menu is entered. NOTE: Each team is given a specific time to use for current and long-term planning. It is strongly recommended that the teams take full advantage of their allotted time to carefully prepare their plans. There is no advantage to an early submission of the yearly plan. The computer will simply make the team wait for the end of the period anyway. It should also be emphasized that a team is penalized for a late submission. Significant dollar reductions from the next year's budget are assessed for each minute the plan is late.

Upon selecting the year-end option from the menu, the teams are asked to confirm that they, indeed, are done for the year. At this point they have the option of returning to the main menu for subsequent adjustments.

When a confirmed entry is made, the year-end module performs several processes in closing out accounting for the year. The first report provided is a summary of utils provided by the weapon systems and dollars spent for the year. This report is immediately displayed at the team's terminal.

Next, the teams are advised of a break period to allow them to relax and collect themselves for the next planning cycle. During the break, the umpire program is called up and each side's data is fed to a central point for recording and determination of the winner for the year. The umpire program also uses a random number generating routine to determine whether a war has occurred during the year. If a war has broken out, messages to that effect are written into files and later retrieved by both sides. The activities of the umpire program are all transparent to the players of the game. Finally, the year-end routine updates the database for the coming year, and passes intelligence data back and forth between the red and blue sides.

When the program becomes active again, an introductory report for the next year will appear on the terminal screen. At this time, the intelligence reports which were requested during the previous year are also available simply by pressing the carriage return key on the terminal keyboard. Following review of the intelligence reports, the teams receive the original menu and begin the process again for the next year.

13. WAR

Throughout the game there is a threat that war could break out in any year. The intelligence sources for each side will keep the teams informed of the best estimate of probability of an outbreak of hostilities for each year.

This information is provided at the beginning of each year and need not be requested or funded. It is also available in the BUDGET module at any time during the year.

If a war occurs, the cost in resources for each team will be reported to them and a winner will be declared. However, the overall winner of the game will be the team with the most net offensive utils for the entire eight-year game period.

14. PENALTIES

If a war occurs both teams will have $400 cut from the budget for next year. In addition, the loser's budget will be cut by an amount equal to twice the difference between the two teams net offensive util count calculated for the year the war occurred.

If a team allocates more funds than it has in its budget during a given year, the budget in the following year will be reduced by twice the amount of the deficit. In addition, as soon as a team's expenditures (allocations plus operating costs) exceed the budget, the program will take it automatically to the year-end module. The team has the option of returning to the main menu. However, with each additional expenditure it will once again be sent to the year-end module. Advisories will be provided to explain what is happening but the awkwardness of this condition will presumably prevent significant overexpenditures.

If a budget is submitted late (that is, if the end-of-year menu selection is not made in the allotted time) the budget for the next year will be reduced by $50 per minute for the first five minutes, and $120 per minute for each minute over five minutes.

If a team does not spend its entire budget for the current year, the remainder of the funds are lost. There is no carryover to the following year.

15. WEAPON SYSTEM EFFECTIVENESS

The total utility of a given weapon syster for one year is not simply the number of utils times the number of systems. The Law of Diminishing Marginal Returns postulates that stockpiling one particular weapon system which has a high utility does not insure success, nor does it support a balanced force concept. Therefore, the final util count for a system in a particular year may be reduced by some amount. The scheme for this util discount is shown in figure A-1. For example, if a team has 100 weapon systems, each with a utility of 55, the team will receive a total utility for that system of 4400 + 300 = 4700 rather than 5500.

16. DOCUMENTATION

Teams will be provided with blank budget allocation forms to assist them in planning where resources will be spent for the year. These forms need not be submitted at any time. They are merely tools to make the organization of the teams easier.

FORCE UNIT UTIL DISCOUNT

Total utils of any individual force unit are discounted as follows:

| TOTAL ORIGINAL UTILS | FINAL NET UTIL VALUES |
|---|---|
| 1 - 2000 | original total |
| 2001 - 3000 | 2000 plus 90% of amount over 2000 |
| 3001 - 4000 | 2900 plus 80% of amount over 3000 |
| 4001 - 5000 | 3700 plus 70% of amount over 4000 |
| 5001 - 6000 | 4400 plus 60% of amount over 5000 |
| 6001 - 7000 | 5000 plus 50% of amount over 6000 |
| 7001 - 8000 | 5500 plus 40% of amount over 7000 |
| 8001 - 9000 | 5900 plus 30% of amount over 8000 |
| 9001 - 10000 | 6200 plus 20% of amount over 9000 |
| 10001 - 11000 | 6400 plus 10% of amount over 10000 |
| over 11000 | 6500 maximum net utils |

Figure A-1. Utility Discount Scheme

Each year both teams receive an actual budget for the current year plus estimated budgets for the next two years. These estimated budgets may change, but they can assist in making some long-range plans. The Multiple Year Defense Program(MYDP) is patterned after the Five Year Defense Program and is provided to help the teams formulate long-term force plans. The MYDP is merely a worksheet and will not be submitted.

Sample forms for the Budget Allocation Form and the Multiple Year Defense Plan are provided in figure A-2 and figure A-3, respectively.

17. UMPIRE

The umpire has only one responsibility in the play of this version of TEMPO. He must signal the start of play when both teams have had the chance to review initial game data.

The umpire can also observe game status in terms of each team's utility totals for each year by logging into 'umpire' with password 'umpire' and examining the file "ump". This is a cumulative file which is updated at the end of each year. Note that the file must be recalled at the end of each year to receive the most current data. A file being displayed will not be automatically updated as it is being viewed.

Year___ Budget_____

OPERATIONS

| SYSTEM | BEGIN YR INVEN | # TO SCRAP | # TO MOD (NOP) | NUMBER TO OPERATE | OPER COST PER UNIT | OPER COST TOTAL | TOTAL UTILS |
|---|---|---|---|---|---|---|---|
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | Total Operating Cost | | |

RESEARCH AND DEVELOPMENT

| System | R&D Cost |
|---|---|
| | |
| | |
| | |
| | |
| | |
| Total R&D Cost | |

ACQUISITION

| System | # to Buy or Mod | Unit Cost | Total Cost |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | Total Acquisition Cost | |

INTELLIGENCE

| Type | Intel Cost | Counter |
|---|---|---|
| Off Force | $100 | $50 |
| Def Force | 100 | 50 |
| Off R&D | 100 | 50 |
| Def R&D | 100 | 50 |
| Subtotals | | |
| Total Intel Cost $ | | |

ANALYSIS

($25 per analysis)

| System | vs | System |
|---|---|---|
| | | |
| | | |
| | | |
| | | |

Total Analysis Cost_____

Figure A-2. Sample Budget Allocation Form

| SYSTEM | Year__ | | Year__ | | Year__ | | Year__ | | Year__ | | Year__ | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | # | $ | # | $ | # | $ | # | $ | # | $ | # | $ |
| OPS | | | | | | | | | | | | |
| ACQ | | | | | | | | | | | | |
| R&D | | | | | | | | | | | | |
| MOD | | | | | | | | | | | | |
| OPS | | | | | | | | | | | | |
| ACQ | | | | | | | | | | | | |
| R&D | | | | | | | | | | | | |
| MOD | | | | | | | | | | | | |
| OPS | | | | | | | | | | | | |
| ACQ | | | | | | | | | | | | |
| R&D | | | | | | | | | | | | |
| MOD | | | | | | | | | | | | |
| OPS | | | | | | | | | | | | |
| ACQ | | | | | | | | | | | | |
| R&D | | | | | | | | | | | | |
| MOD | | | | | | | | | | | | |
| OPS | | | | | | | | | | | | |
| ACQ | | | | | | | | | | | | |
| R&D | | | | | | | | | | | | |
| MOD | | | | | | | | | | | | |
| Totals Est Budget | | | | | | | | | | | | |
| Difference | | | | | | | | | | | | |

Figure A-3. Sample Multiple Year Defense Program Form

```
/*****************************************************************
* This program is the primary source for the TEMPO military*
* planning game in the Naval Postgraduate School Command,  *
* Control, and Communications Laboratory. This is the code *
* for the Blue Force.  The Red Force code is identical     *
* except for the interface with the umpire program. This   *
* variation by described at the end of the Blue Force      *
* Program.  This program is called BLUSIDE.                *
*****************************************************************/

float x;
double fmod();
int atoi(),rand(),srand(),hrstrt,minstrt,quit,pw,pw1;
int tvec[2],time(),i,t,hleft,mleft,tdisp,timebst;
int *localtime(),*locl,clock[3];
int oa1[11]{0,40,60,40,25,30,0,0,0,90,0};
int oa1m[11]{0,0,150,80,80,0,200,0,0,0,1};
int oa2[11]{0,0,200,50,200,0,500,400,700,0,3};
int ob1[11]{0,20,80,60,40,20,0,0,0,30,0};
int ob2[11]{0,0,100,90,250,0,400,600,400,0,3};
int da1[11]{0,20,100,60,50,30,0,0,0,0,0};
int db1[11]{0,50,50,40,20,30,0,0,0,0,0};
int db2[11]{0,0,100,30,100,0,700,700,0,0,2};
int oa3[10],ob1m[10],ob3[10];
int ob4[10],da2[10],da3[10];
int cost,blufb,loss,upay,acost,acostt,mode,init;
int db3[10],db4[10];
int year,n,funds,bucks,qty,trig,tempflg[15];
int z,flg[15],stop,intflag[10],rdyr[11],tcost,tutil,apprp;
int offa,offb,defa,defb,ops1,ops2,ops3,ops4,opcost,spent;
int buy[17],scrap[17],bmcd,mmcd,bmon,mmon,iwar;
int send,timeout,warrep,warmsg,deficit,app1,app2;
char cont,sel,sct,war[5],abuf[10],pur,clear 014;
char str[5],anal[3];
char str2[] "\n
```

```
                    1 - CURRENT FORCE STATUS AND DATA\n\n
                    2 - RESEARCH AND DEVELOPMENT DATA\n\n
                    3 - CURRENT BUDGET DATA\n\n
                    4 - BUY ADDITIONAL SYSTEMS\n\n
                    5 - MODIFY SYSTEMS IN CURRENT INVENTORY\n\n
                    6 - SCRAP SYSTEMS IN CURRENT INVENTORY\n\n
                    7 - BUY INTELLIGENCE DATA OR COUNTERINTELLIGENCE\n\n
                    8 - PERFORM TRADE-OFF ANALYSES BETWEEN PROPOSED SYSTEMS\n\n
                    9 - RECOVERY FROM ENTRY ERRORS\n\n
                    0 - SUBMIT COMPLETED PLANNING AND BUDGETING FOR THIS YEAR\n\n
                    T - TIME UPDATE\n\n\n\n
                    PLEASE SELECT A CHOICE FROM THE MENU.";

/* BEGIN MAIN PROGRAM */
main(argc,argv)

   int argc;
   int **argv;
   {
   int *ptr;
   int seed;
   ptr = *argv;
   mode = *ptr;

/*INITIALIZE THE CLOCK AND START ELAPSED TIME */
   time(tvec);
   locl = localtime(tvec);
   for(i=0;i<3;i++) clock[i]=locl[i];
   hrstrt=clock[2];
   minstrt=clock[1];
   seed = locl[0];
   srand(seed);

/*INTIALIZE FLAGS AND TRIGGERS */
   quit=0;
   year = 1;       timeout=0;
   for(i=1;i<11;i++) rdyr[i]=0;
```

```
for(i=1;i<13;i++) flg[i]=2;
for(i=1;i<12;i++) tempflg[i]=0;
for(i=1;i<9;i++)intflag[i]=0;

init = open("/usr/umpire/blusum",2);
x=0;
itoa(x);
write(init,str,5);
close(init);

oa3[1]=0;       ob3[1]=0;       da2[1]=0;       da3[1]=0;
db3[1]=0;       ob1m[1]=0;      cb4[1]=0;
database();

while(quit==0){
   newpge();
   printf("%s",str2);
   sel=getans();
   switch(sel){
      case '1':
         inform();
         break;
      case '2':
         research();
         break;
      case '3':
         budget();
         break;
      case '4':
         buy1();
         break;
      case '5':
         modify();
         break;
      case '6':
         scrap1();
         break;
```

```
            case '7':
               intel();
               break;
            case '8':
               analyze();
               break;
            case '9':
               reinit();
               break;
            case '0':
               done();
               break;
            case 'T': case't':
               break;
            default:
               printf("\n\nYOUR ONLY CHOICES ARE SINGLE DIGIT INTEGERS,AND TIME.\
                         \nPLEASE RESELECT.(1 through 0 or T) \n");
                           sleep(2);
                           break;
                    }
         }
}

/*********************************************
   This subroutine calculates elapsed time and
   projects the time remaining on the screen
   with each display.                        */

gclock()
   {
   time(tvec);
   locl = localtime(tvec);
   for(i=0;i<3;i++) clock[i]=locl[i];
   if(clock[1]<10)
      printf(" %d:0%d ",clock[2],clock[1]);
   else printf(" %d:%d ",clock[2],clock[1]);
   if(clock[1]<minstrt){
```

```
            clock[2]=clock[2]-1;
            clock[1]=clock[1]+60;
            }
         hleft=clock[2]-hrstrt;
         mleft=clock[1]-minstrt;
         if(mode==1){
            if(year==1) t=35;
            else if(year==2) t=20;
            else if(year==3) t=15;
            else if(year==4) t=10;
            else if(year>=5) t=20;
            }
         else{
            if(year==1) t=50;
            else if(year==2) t=45;
            else if(year==3) t=35;
            else if(year==4) t=30;
            else if(year>=5) t=35;
            }
         tdisp = t-mleft;
         printf("   YOU HAVE %d MINUTES LEFT TO COMPLETE THIS YEAR'S PLANNING.\
         \n\n\n\n\n",tdisp);
      }

      /*****************************************************
         The INFORM subroutine presents current force status
         to the team.  This data includes current active in-
         ventory, acquisition cost, operation cost, utility
         for each system, purchase limits, and total system
         utility and operation cost for the total force of
         each system.                                      */

      inform()
         {
         newpge();
         printf(" THIS IS YEAR %d OF THE GAME.\n\n",year);
         printf("\n THE SYSTEMS YOU HAVE IN YOUR PRESENT INVENTORY AND\
```

```
      \nTHEIR ASSOCIATED DATA ARE DETAILED IN THE TABLE BELOW:\n\n");
   printf("                                                    PURCH");
   printf("          TOTAL\
     \nSYSTEM  INVENTORY  AQCOST  OPCOST    UTILS    LIMIT");
   printf("       OPCOST   UTILS\n");
   if(oa1[1]>=1) buff();
   if(oa1m[1]>=1)buffmod();
   if(oa2[1]>=1)fcneoneone();
   if(oa3[1]>=1)stealth();
   if(ob1[1]>=1)minute();
   if(ob1m[1]>=1)minmod();
   if(ob2[1]>=1)alcm();
   if(ob3[1]>=1)mx();
   if(ob4[1]>=1)fcb();
   if(da1[1]>=1)chap();
   if(da2[1]>=1)hawk();
   if(da3[1]>=1)patriot();
   if(db1[1]>=1)sprint();
   if(db2[1]>=1)spartan();
   if(db3[1]>=1)loads();
   if(db4[1]>=1)vader();
   printf("\n\n\n\n\n\nTYPE CARRIAGE RETURN TO RETURN TO MAIN MENU");
   cont=getchar();
   return;
}

/*****************************************************
   The RESEARCH subroutine is used to display the
   systems which are available for R&D, to give the
   pertinent R&D data for each system and to allow
   the players to invest in R&D programs.        */

research()
   {
   newpge();
   printf("THE SYSTEMS WHICH YOU MAY BEGIN TO R&D THIS YEAR ARE:\n\n");
   switch(year){
```

```
        case 1: printf("\n
                        OFFENSIVE AVIONICS UPGRADE FOR B-52\n
                        FB-111H BOMBER\n
                        AIR LAUNCHED CRUISE MISSILE(ALCM)\n
                        SPARTAN ABM\n\n");
                        break;

        case 2: printf("\n
                        OFFENSIVE AVIONICS UPGRADE FOR B-52\n
                        MINUTEMAN III GUIDANCE UPGRADE\n
                        MX MOBILE ICBM\n
                        IMPROVED HAWK SAM\n
                        LoADS ABM\n\n");
                        break;

        case 3: printf("\n
                        OFFENSIVE AVIONICS UPGRADE FOR B-52\n
                        MINUTEMAN III GUIDANCE UPGRADE\n
                        B-1 BOMBER\n
                        FOB SPACE DIRECTED ICBM\n
                        PATRIOT SAM\n
                        VADER ABM\n\n");
                        break;
        default:printf("\n
                        NONE.   YOU HAVE EXHAUSTED THE HORIZONS OF\n
                        NEW TECHNOLOGY AND MUST NOW RELY ON YOUR \n
                        PAST JUDGEMENTS AS TO WHAT WAS WORTH DEVELOPING.");
                        break;
        }
printf("\n\n\nDO YOU WISH TO SEE A COMPLETE R&D REPORT FOR THIS YEAR? (y/n)");
   yesno();
   switch(sct){
      case 'Y': case 'y':
         if(flg[2]==1) oa2[6]=500;
         if(flg[3]==1) oa3[6]=800;
         if(flg[5]==1) ob2[6]=400;
         if(flg[6]==1) ob3[6]=300;
```

```
            if(flg[7]==1) ob4[6]=800;
            if(flg[8]==1) da2[6]=300;
            if(flg[9]==1) da3[6]=1000;
            if(flg[10]==1) db2[6]=700;
            if(flg[11]==1) db3[6]=1000;
            if(flg[12]==1) db4[6]=999;
            newpge();
printf("                                        *****R&D COSTS******** YR TILL\
      \nSYSTEM  AQCOST  OPCOST  UTILS  1ST YR   2ND YR  3RD YR   AVAIL\n");
            if(oa1m[6]>=1 && flg[1]!=1)buffmod();
            if(oa2[6]>=1 && rdyr[1]<3)foneoneone();
            if(oa3[6]>=1 && rdyr[2]<3)stealth();
            if(ob1m[6]>=1 && flg[4]!=1)minmod();
            if(ob2[6]>=1 && rdyr[3]<3)alcm();
            if(ob3[6]>=1 && rdyr[4]<3)mx();
            if(ob4[6]>=1 && rdyr[5]<2)fob();
            if(da2[6]>=1 && rdyr[6]<3)hawk();
            if(da3[6]>=1 && rdyr[7]<3)patriot();
            if(db2[6]>=1 && rdyr[8]<2)spartan();
            if(db3[6]>=1 && flg[11]!=1)loads();
            if(db4[6]>=1 && rdyr[10]<2)vader();
            break;
         case 'N': case 'n':
            break;
         }
      printf("\n\n\n\n\n\nDO YOU WISH TO INVEST R&D FUNDS IN ANY OF ");
      printf("THESE PROJECTS? (y/n) ");
      yesno();
      newpge();
      switch(sct){
         case 'Y': case 'y':
      printf("                                  R&D  COSTS            STATUS OF\
            \n                              1ST   2ND   3RD       ONGOING PROGRAMS.\
            \n                              YEAR  YEAR  YEAR       HAVE COMPLETED:\n\n");
        if(oa1m[6]>0 && flg[1]!=1)
      printf("B - UPGRADED B-52G       %d    000    000",oa1m[6]);
        if(oa2[6]>0 && rdyr[1]<3){
```

```
printf("\nC - FB-111H BOMBER      %d    %d    %d",oa2[6],oa2[7],oa2[8]);
if(flg[2]==1)
printf("        %d YR OF 3 YEAR PROGRAM",rdyr[1]);}
  if(oa3[6]>0 && rdyr[2]<3){
printf("\nD - B-1 BOMBER          %d    %d    %d",oa3[6],oa3[7],oa3[8]);
if(flg[3]==1)
printf("        %d YR OF 3 YEAR PROGRAM",rdyr[2]);}
  if(ob1m[6]>0 && flg[4]!=1)
printf("\nF - UPGRADED MM-III     %d    000    000",ob1m[6]);
  if(ob2[6]>0 && rdyr[3]<3){
printf("\nG - ALCM                %d    %d    %d",ob2[6],ob2[7],ob2[8]);
if(flg[5]==1)
printf("        %d YR OF 3 YEAR PROGRAM",rdyr[3]);}
  if(ob3[6]>0 && rdyr[4]<3){
printf("\nH - MX MOBILE ICBM      %d    %d    %d",ob3[6],ob3[7],ob3[8]);
if(flg[6]==1)
printf("        %d YR OF 3 YEAR PROGRAM",rdyr[4]);}
  if(ob4[6]>0 && rdyr[5]<2){
printf("\nI - FOB                 %d    %d    %d",ob4[6],ob4[7],ob4[8]);
if(flg[7]==1)
printf("        %d YR OF 2 YEAR PROGRAM",rdyr[5]);}
  if(da2[6]>0 && rdyr[6]<3){
printf("\nK - HAWK SAM            %d    %d    %d",da2[6],da2[7],da2[8]);
if(flg[8]==1)
printf("        %d YR OF 3 YEAR PROGRAM",rdyr[6]);}
  if(da3[6]>0 && rdyr[7]<3){
printf("\nL - PATRIOT SAM         %d    %d    %d",da3[6],da3[7],da3[8]);
if(flg[9]==1)
printf("        %d YR OF 3 YEAR PROGRAM",rdyr[7]);}
  if(db2[6]>0 && rdyr[8]<2){
printf("\nN - SPARTAN ABM         %d    %d    %d",db2[6],db2[7],db2[8]);
if(flg[10]==1)
printf("        %d YR OF 2 YEAR PROGRAM",rdyr[8]);}
  if(db3[6]>0 && rdyr[9]<1){
printf("\nP - LoADS ABM           %d    %d    %d",db3[6],db3[7],db3[8]);
if(flg[11]==1)
printf("        %d YR OF 1 YEAR PROGRAM",rdyr[9]);}
```

```
  if(db4[6]>0 && rdyr[10]<2){
printf("\nQ - VADER ABM            %d     %d     %d",db4[6],db4[7],db4[8]);
if(flg[12]==1)
printf("        %d YR OF 2 YEAR PROGRAM",rdyr[10]);}

printf("\n\nNOTE: YOU MUST CONTINUE TO FUND THE PROGRAMS LISTED AS IN \
   \nPROCESS IN ORDER TO COMPLETE THEM ON SCHEDULE.\n");

printf("\nSELECT THE SYSTEM YOU WISH TO R&D BY TYPING ITS CORRESPONDING");
printf(" LETTER.\n");
pur=getans();
trig=1;
while(trig==1){
   switch(pur){
      case'B':case'b':
         if(flg[1]==1){
            complete();
            break;
            }
         else{
            flg[1]=1;
            tempflg[1]=1;
            funds=funds-oa1m[6];
            oa1m[5]=30;
            left();
            }
         break;

      case'C':case'c':
         if(tempflg[2]==1){nicetry(); break;}
         if((year>=2)&&(flg[2]!=1)) sorry();
         else {
            rdyr[1]=rdyr[1]+1;
            if(rdyr[1]<=3){
       printf("R&D COST FOR THE FB-111H THIS YEAR IS %d\n",oa2[rdyr[1]+5]);
               flg[2]=1;
               tempflg[2]=1;
```

```
            funds=funds-oa2[rdyr[1]+5];
            if(rdyr[1]==3) oa2[5]=15;
            left();
          }
          else
             complete();
       }
       break;

case'D':case'd':
   if(tempflg[3]==1){nicetry(); break;}
   if(year<=2) notavail();
   else if((year>=4)&&(flg[3]!=1)) sorry();
   else{
      rdyr[2]=rdyr[2]+1;
      if(rdyr[2]<=3){
    printf("R&D COST FOR THE B-1 THIS YEAR IS %d\n",oa3[rdyr[2]+5]);
         flg[3]=1;
         tempflg[3]=1;
         funds=funds-oa3[rdyr[2]+5];
         if(rdyr[2]==3) oa3[5]=15;
         left();
         }
       else
          complete();
       }
   break;

case'F':case'f':
   if(tempflg[4]==1){nicetry(); break;}
   if(year<2) notavail();
   else if(flg[4]==1){
      complete();
      break;
      }
   else{
      flg[4]=1;
```

```
            tempflg[4]=1;
            funds=funds-ob1m[6];
            ob1m[5]=20;
            left();
            }
         break;

      case'G':case'g':
         if(tempflg[5]==1){nicetry(); break;}
         if((year>=2)&&(flg[5]!=1)) sorry();
         else{
            rdyr[3]=rdyr[3]+1;
            if(rdyr[3]<=3){
          printf("R&D COST FOR THE ALCM THIS YEAR IS %d\n",ob2[rdyr[3]+5]);
               flg[5]=1;
               tempflg[5]=1;
               funds=funds-ob2[rdyr[3]+5];
               if(rdyr[3]==3) ob2[5]=15;
               left();
               }
            else
               complete();
            }
         break;

      case'H':case'h':
         if(tempflg[6]==1){nicetry(); break;}
         if(year==1) notavail();
         else if((year>=3)&&(flg[6]!=1)) sorry();
         else{
            rdyr[4]=rdyr[4]+1;
            if(rdyr[4]<=3){
            printf("R&D COST FOR THE MX THIS YEAR IS %d\n",ob3[rdyr[4]+5]);
               flg[6]=1;
               tempflg[6]=1;
               funds=funds-ob3[rdyr[4]+5];
               if(rdyr[4]==3) ob3[5]=15;
```

```
                left();
                }
             else
                complete();
             }
          break;

       case'I':case'i':
          if(tempflg[7]==1){nicetry(); break;}
          if(year<=2) notavail();
          else if((year>=4)&&(flg[7]!=1)) sorry();
          else{
             rdyr[5]=rdyr[5]+1;
             if(rdyr[5]<=2){
            printf("R&D COST FOR THE FOB THIS YEAR IS %d\n",ob4[rdyr[5]+5]);
                flg[7]=1;
                tempflg[7]=1;
                funds=funds-ob4[rdyr[5]+5];
                if(rdyr[5]==2) ob4[5]=15;
                left();
                }
             else
                complete();
             }
          break;

       case'K':case'k':
          if(tempflg[8]==1){nicetry(); break;}
          if(year==1) notavail();
          else if((year>=3)&&(flg[8]!=1)) sorry();
          else{
             rdyr[6]=rdyr[6]+1;
             if(rdyr[6]<=3){
printf("R&D COST FOR THE IMPROVED HAWK THIS YEAR IS %d\n",da2[rdyr[6]+5]);
                flg[8]=1;
                tempflg[8]=1;
                funds=funds-da2[rdyr[6]+5];
```

```
            if(rdyr[6]==3) da2[5]=15;
            left();
            }
         else
            complete();
         }
      break;

   case'L':case'l':
      if(tempflg[9]==1){nicetry(); break;}
      if(year<=2) notavail();
      else if((year>=4)&&(flg[9]!=1)) sorry();
      else{
         rdyr[7]=rdyr[7]+1;
         if(rdyr[7]<=3){
    printf("R&D COST FOR THE PATRIOT THIS YEAR IS %d\n",da3[rdyr[7]+5]);
            flg[9]=1;
            tempflg[9]=1;
            funds=funds-da3[rdyr[7]+5];
            if(rdyr[7]==3) da3[5]=15;
            left();
            }
         else
            complete();
         }
      break;

   case'N':case'n':
      if(tempflg[10]==1){nicetry(); break;}
      if((year>=2)&&(flg[10]!=1)) sorry();
      else{
         rdyr[8]=rdyr[8]+1;
         if(rdyr[8]<=2){
    printf("R&D COST FOR THE SPARTAN THIS YEAR IS %d\n",db2[rdyr[8]+5]);
            flg[10]=1;
            tempflg[10]=1;
            funds=funds-db2[rdyr[8]+5];
```

```
                if(rdyr[8]==2) db2[5]=15;
                left();
                }
             else
                complete();
             }
          break;

       case'P':case'p':
          if(tempflg[11]==1){nicetry(); break;}
          if(year==1) notavail();
          else if((year>=3)&&(flg[11]!=1)) sorry();
          else{
             rdyr[9]=rdyr[9]+1;
             if(rdyr[9]<=1){
    printf("R&D COST FOR THE LoADS ABM THIS YEAR IS %d\n",db3[rdyr[9]+5]);
             flg[11]=1;
             tempflg[11]=1;
             funds=funds-db3[rdyr[9]+5];
             if(rdyr[9]==1) db3[5]=15;
             left();
             }
          else
             complete();
          }
          break;

       case'Q':case'q':
          if(tempflg[12]==1){nicetry(); break;}
          if(year<=2) notavail();
          else if((year>=4)&&(flg[12]!=1)) sorry();
          else{
             rdyr[10]=rdyr[10]+1;
             if(rdyr[10]<=2){
    printf("R&D COST FOR THE VADER ABM THIS YEAR IS %d\n",db4[rdyr[10]+5]);
             flg[12]=1;
             tempflg[12]=1;
```

```
                        funds=funds-db4[rdyr[10]+5];
                        if(rdyr[10]==2) db4[5]=15;
                        left();
                        }
                     else
                        complete();
                     }
                  break;

               default:
                  printf("YOUR INPUT WAS NOT A VALID CHOICE.");
               }

            if(timeout==1){done(); trig=0; return;}
            printf("\n\nDO YOU WANT ANY OTHER SYSTEMS? (y/n)  ");
            which();
            }
            break;
      }
      return;
   }
   /*****************************************************
      The next eight subroutines are utilities or advisories
      called by RESEARCH and/or BUY1 . NICETRY is called when
      the team tries to purchase R&D on the same system more
      than once in the same year.                        */

   nicetry(){
      printf("MORE FUNDS POURED ON THIS PROGRAM THIS YEAR WILL DO NO\
         \nGOOD. YOU MUST WAIT UNTIL NEXT YEAR.\n");
      }
   /*****************************************************
      NOTAVAIL is called when the team tries to R&D a system
      which has not yet been offered to them.            */

   notavail(){
      printf("THIS TECHNOLOGY IS NOT YET AVAILABLE.\n");
```

```
   return;
   }
/*************************************************************
   MUSTFUND is called when the team tries to purchase a
   system before the R&D is complete.                        */

mustfund(){
   printf("YOU MUST FUND THE R&D BEFORE YOU CAN PURCHASE THE ");
   return;
   }
/*************************************************************
   LEFT is called after every transaction in which budget
   monies are spent. It updates the current total of
   yearly expenditures and checks to see if the total of
   opcost and expenditures exceeds the budget for the
   year.  If so, it sets a flag which dumps the user in the
   end of year routine(DONE) the next time they hit the
   main menu.                                                */

left(){
   costops();
   bucks=apprp-funds;
   if(apprp-cpcost-bucks<0){
      timeout=1;
      printf("\n\nCONGRATULATIONS! YOU HAVE JUST WIPED OUT\
         \nYOUR ENTIRE BANKROLL FOR THIS YEAR. YOU WILL\
         \nBE AUTOMATICALLY TRANSPORTED TO THE END-OF-YEAR \
         \nMODULE AS A SUBTLE HINT THAT YOU'VE DONE ENOUGH FOR\
         \nONE YEAR.(Type CR)");
      cont=getchar();
      }
   else
      printf("YOU NOW HAVE $%d LEFT THIS YEAR.\
         \n(This includes your O&M money)\n",funds);
   return;
}
/*************************************************************
```

```
   INPROGRESS is called when the team tries to buy a system
   which has begun R&D but is not yet done.                */

inprogress(){
   printf("YOUR R&D PROGRAM IS NOT YET COMPLETE FOR THIS SYSTEM");
   return;
}
/*************************************************************
   COMPLETE is called when the team tries to invest R&D
   funds in a system which is already in the inventory.*/

complete(){
           printf("YOUR R&D ON THIS SYSTEM IS COMPLETE");
           return;
}
/*************************************************************
   SORRY is called when the team tries to invest R&D funds
   in a system which they did not pick up the first year
   it was offered. There are no second chances.          */

sorry(){
printf("SORRY BUT YOU MISSED YOUR CHANCE TO BRING THIS SYSTEM ON BOARD.\n\n");
   return;
}
/*************************************************************
   TCTU computes total cost and total utility to be displayed
   with the INFORM output.                                  */

tctu(q1,p1,u1){
   tcost=q1*p1;
   tutil=q1*u1;
   printf("    %d     %d",tcost,tutil);
   return;
}
/*************************************************************
   The next series of utility subroutine are used to write
   both the INFORM chart and the RESEARCH data chart. They
```

```
   essentially just read out the database parameters in the
   order they are presented.                                  */

buff(){
  printf("\nB-52G      ");
  for(i=1;i<=5;i=i+1){
     if(oa1[i]==0)
        printf(" 000       ");
     else
        printf(" %d       ",oa1[i]);
     }
  tctu(oa1[1],oa1[3],oa1[4]);
  return;
}

buffmod(){
   printf("\nB-52G    \n");
   printf("(MOD)      ");
   switch(sel){
      case '1':
         for(i=1;i<=5;i=i+1){
            if(oa1m[i]==0)
               printf(" 000       ");
            else
               printf(" %d       ",oa1m[i]);
            }
         tctu(oa1m[1],oa1m[3],oa1m[4]);
         break;
      case '2':
         for(i=2;i<=10;i++)
            if((i!=5)&&(i!=9)){
               if(oa1m[i]==0)
                  printf(" 000     ");
               else
                  printf(" %d     ",oa1m[i]);
               }
            break;
```

```
        }
    return;
}

foneoneone(){
    printf("\nFB-111H    ");
    switch(sel){
        case '1': for(i=1;i<=5;i=i+1){
                    if(oa2[i]==0)
                        printf(" 000       ");
                    else
                        printf(" %d       ",oa2[i]);
                    }
                  tctu(oa2[1],oa2[3],oa2[4]);
                  break;
        case '2': for(i=2;i<=10;i++)
                    if((i!=5)&&(i!=9)){
                        if(oa2[i]==0)
                            printf(" 000     ");
                        else
                            printf(" %d     ",oa2[i]);
                        }
                  break;
        }
    return;
}

stealth(){
    printf("\nB-1        ");
    switch(sel){
        case '1': for(i=1;i<=5;i=i+1){
                    if(oa3[i]==0)
                        printf(" 00       ");
                    else
                        printf(" %d       ",oa3[i]);
                  }
                  tctu(oa3[1],oa3[3],oa3[4]);
```

```
                    break;
        case '2': for(i=2;i<=10;i++)
                    if((i!=5)&&(i!=9)){
                       if(oa3[i]==0)
                          printf(" 000     ");
                       else
                          printf(" %d     ",oa3[i]);
                       }
                    break;
                    }
    return;
}

minute(){
    printf("\nMM-III     ");
    for(i=1;i<=5;i=i+1){
       if(ob1[i]==0)
          printf(" 000       ");
       else
          printf(" %d       ",ob1[i]);
    }
    tctu(ob1[1],ob1[3],ob1[4]);
    return;
}

minmod(){
    printf("\nMM-III     ");
    printf("\n(MOD)      ");
    switch(sel){
       case '1': for(i=1;i<=5;i=i+1){
                    if(ob1m[i]==0)
                       printf(" 000       ");
                    else
                       printf(" %d       ",ob1m[i]);
                 }
                 tctu(ob1m[1],ob1m[3],ob1m[4]);
                 break;
```

```
      case '2': for(i=2;i<=10;i++)
                   if((i!=5)&&(i!=9)){
                      if(ob1m[i]==0)
                         printf(" 000    ");
                      else
                         printf(" %d    ",ob1m[i]);
                   }
                break;
       }
    return;
}

alcm(){
   printf("\nALCM      ");
   switch(sel){
      case '1': for(i=1;i<=5;i=i+1){
                   if(ob2[i]==0)
                      printf(" 000      ");
                   else
                      printf(" %d      ",ob2[i]);
                }
                tctu(ob2[1],ob2[3],ob2[4]);
                break;
      case '2': for(i=2;i<=10;i++)
                   if((i!=5)&&(i!=9)){
                      if(ob2[i]==0)
                         printf(" 000    ");
                      else
                         printf(" %d    ",ob2[i]);
                   }
                break;
       }
    return;
}

mx(){
   printf("\nMX        ");
```

```
    switch(sel){
       case '1': for(i=1;i<=5;i=i+1){
                    if(ob3[i]==0)
                       printf(" 000      ");
                    else
                       printf(" %d       ",ob3[i]);
                 }
                 tctu(ob3[1],ob3[3],ob3[4]);
                 break;
       case '2': for(i=2;i<=10;i++)
                    if((i!=5)&&(i!=9)) {
                       if(ob3[i]==0)
                          printf(" 000    ");
                       else
                          printf(" %d     ",ob3[i]);
                     }
                  break;
       }
    return;
}

fob(){
    printf("\nFOB       ");
    switch(sel){
       case '1': for(i=1;i<=5;i=i+1){
                    if(ob4[i]==0)
                       printf(" 000      ");
                    else
                       printf(" %d       ",ob4[i]);
                 }
                 tctu(ob4[1],ob4[3],ob4[4]);
                 break;
       case '2': for(i=2;i<=10;i++)
                    if((i!=5)&&(i!=9)){
                       if(ob4[i]==0)
                          printf(" 000    ");
                       else
```

```
                        printf(" %d     ",ob4[i]);
                    }
                break;
        }
    return;
}

chap(){
    printf("\nCHAPPARAL ");
    for(i=1;i<=5;i=i+1){
        if(da1[i]==0)
            printf(" 000       ");
        else
            printf(" %d       ",da1[i]);
    }
    tctu(da1[1],da1[3],da1[4]);
    return;
}

hawk(){
    printf("\nHAWK      ");
    switch(sel){
        case '1': for(i=1;i<=5;i=i+1){
                    if(da2[i]==0)
                        printf(" 000       ");
                    else
                        printf(" %d       ",da2[i]);
                }
                tctu(da2[1],da2[3],da2[4]);
                break;
        case '2': for(i=2;i<=10;i++)
                    if((i!=5)&&(i!=9)){
                        if(da2[i]==0)
                            printf(" 000     ");
                        else
                            printf(" %d     ",da2[i]);
                    }
```

```
                    break;
            }
        return;
    }
    patriot(){
        printf("\nPATRIOT .  ");
        switch(sel){
            case '1': for(i=1;i<=5;i=i+1){
                        if(da3[i]==0)
                            printf(" 000      ");
                        else
                            printf(" %d      ",da3[i]);
                      }
                      tctu(da3[1],da3[3],da3[4]);
                      break;
            case '2': for(i=2;i<=10;i++)
                        if((i!=5)&&(i!=9)){
                            if(da3[i]==0)
                                printf(" 000    ");
                            else
                                printf(" %d    ",da3[i]);
                      }
                    break;
            }
        return;
    }
    sprint(){
        printf("\nSPRINT    ");
        for(i=1;i<=5;i=i+1){
            if(db1[i]==0)
                printf(" 000      ");
            else
                printf(" %d      ",db1[i]);
        }
        tctu(db1[1],db1[3],db1[4]);
        return;
    }
```

```
spartan(){
   printf("\nSPARTAN   ");
   switch(sel){
      case '1': for(i=1;i<=5;i=i+1){
                   if(db2[i]==0)
                      printf(" 000      ");
                   else
                      printf(" %d       ",db2[i]);
                }
                tctu(db2[1],db2[3],db2[4]);
                break;
      case '2': for(i=2;i<=10;i++)
                   if((i!=5)&&(i!=9)){
                      if(db2[i]==0)
                         printf(" 000    ");
                      else
                         printf(" %d     ",db2[i]);
                    }
                 break;
      }
   return;
}

loads(){
   printf("\nLoADS     ");
   switch(sel){
      case '1': for(i=1;i<=5;i=i+1){
                   if(db3[i]==0)
                      printf(" 000      ");
                   else
                      printf(" %d       ",db3[i]);
                }
                tctu(db3[1],db3[3],db3[4]);
                break;
      case '2': for(i=2;i<=10;i++)
                   if((i!=5)&&(i!=9)){
                      if(db3[i]==0)
```

```
                        printf(" 000     ");
                     else
                        printf(" %d     ",db3[i]);
                  }
               break;
         }
      return;
   }

   vader(){
      printf("\nVADER     ");
      switch(sel){
         case '1': for(i=1;i<=5;i=i+1){
                     if(db4[i]==0)
                        printf(" 000      ");
                     else
                        printf(" %d      ",db4[i]);
                  }
                  tctu(db4[1],db4[3],db4[4]);
                  break;
         case '2': for(i=2;i<=10;i++)
                     if((i!=5)&&(i!=9)){
                        if(db4[i]==0)
                           printf(" 000     ");
                        else
                           printf(" %d     ",db4[i]);
                     }
                  break;
         }
      return;
   }
   /*****************************************************
     The BUDGET subroutine provides the teams with a review
     of their budget for the year and estimates for the next
     two years.  It also provides a running total of dollars
     spent up to this point in the year and the total oper-
     ating costs which will be required for the year.  The
```

```
  probabilities of war for the current year and the next
  year are also available in this module.                 */

budget()
   {
   newpge();
   printf("\n\n\n\nYOUR BUDGET FOR YEAR %d IS $%d .\n\n",year,apprp);
   printf("BUDGET ESTIMATE FOR NEXT YEAR IS $%d.\n",app1);
   printf("BUDGET ESTIMATE FOR THE FOLLOWING YEAR IS $%d.\n\n",app2);
   bucks=apprp-funds;
   printf(" YOU HAVE ALLOCATED %d DOLLARS SO FAR THIS YEAR.\n",bucks);
   costops();
   printf("\n\nKEEP IN MIND THAT THE COST TO OPERATE YOUR CURRENT\
      \nFORCES THIS YEAR WILL BE %d.",opcost);
   printf("\n\n\n\nPROBABILITY OF WAR THIS YEAR IS 0.%d.",pw);
   printf("\nESTIMATED PROBABILITY OF WAR FOR NEXT YEAR IS 0.%d.",pw1);
   printf("\n\n\n\n\n\n\n\n\nTYPE CARRIAGE RETURN FOR MAIN MENU\n\n");
   cont=getchar();
   return;
}
/*************************************************
  BUY1 is the subroutine which allows the teams
  to buy new weapon systems for the inventory.
  It calls the utility subroutines described
  above for display of data on systems avail-
  able.                                          */

buy1()
   {
   newpge();
   printf("THE SYSTEMS AVAILABLE FOR YOUR PURCHASE ARE:\n\n");
   printf("                                    PURCH\n");
   printf("                  AQCOST  UTILS  LIMIT\n\n");
   if(oa1[5]>0)
     printf("A - B-52G BOMBER     %d    %d      %d\n\n",oa1[2],oa1[4],oa1[5]);
   if(oa1m[5]>0)
  printf("B - UPGRADED B-52G   %d    %d      %d\n\n",oa1m[2],oa1m[4],oa1m[5]);
```

```
 if(oa2[5]>0)
   printf("C - FB-111H BOMBER    %d     %d       %d\n\n",oa2[2],oa2[4],oa2[5]);
 if(oa3[5]>0)
   printf("D - B-1 BOMBER        %d     %d       %d\n\n",oa3[2],oa3[4],oa3[5]);
 if(ob1[5]>0)
   printf("E - MINUTEMAN III     %d     %d       %d\n\n",ob1[2],ob1[4],ob1[5]);
 if(ob1m[5]>0)
printf("F - UPGRADED MM-III  %d     %d       %d\n\n",ob1m[2],ob1m[4],ob1m[5]);
 if(ob2[5]>0)
   printf("G - ALCM              %d     %d       %d\n\n",ob2[2],ob2[4],ob2[5]);
 if(ob3[5]>0)
   printf("H - MX MOBILE ICBM    %d     %d       %d\n\n",ob3[2],ob3[4],ob3[5]);
 if(ob4[5]>0)
   printf("I - FOB               %d     %d       %d\n\n",ob4[2],ob4[4],ob4[5]);
 if(da1[5]>0)
   printf("J - CHAPPARAL SAM     %d     %d       %d\n\n",da1[2],da1[4],da1[5]);
 if(da2[5]>0)
   printf("K - HAWK SAM          %d     %d       %d\n\n",da2[2],da2[4],da2[5]);
 if(da3[5]>0)
   printf("L - PATRIOT SAM       %d     %d       %d\n\n",da3[2],da3[4],da3[5]);
 if(db1[5]>0)
   printf("M - SPRINT ABM        %d     %d       %d\n\n",db1[2],db1[4],db1[5]);
 if(db2[5]>0)
   printf("N - SPARTAN ABM       %d     %d       %d\n\n",db2[2],db2[4],db2[5]);
 if(db3[5]>0)
   printf("P - LoADS ABM         %d     %d       %d\n\n",db3[2],db3[4],db3[5]);
 if(db4[5]>0)
   printf("Q - VADER ABM         %d     %d       %d\n\n",db4[2],db4[4],db4[5]);


 printf("SELECT THE SYSTEM YOU WISH TO BUY BY TYPING ITS CORRESPONDING");
 printf(" LETTER.\n");
 pur=getans();
 trig=1;
 while(trig==1){
     switch(pur){
        case'A': case'a':
```

```
        buy[1]=buysub(oa1[2],oa1[5]);
        break;
    case'B': case'b':
        if(flg[1]!=1) {mustfund(); printf("B-52G MOD.");}
        else buy[2]=buysub(oa1m[2],oa1m[5]);
        break;
    case'C': case'c':
        if(year==1 && flg[2]!=1) {mustfund(); printf("FB-111H.");}
        else if(year>1 && flg[2]!=1) sorry();
        else if(rdyr[1]<3 && flg[2]==1) inprogress();
        else {oa2[5]=15;
              buy[3]=buysub(oa2[2],oa2[5]);}
        break;
    case'D': case'd':
        if(year==1) notavail();
        else if(year==2 && flg[3]!=1) {mustfund(); printf("B-1.");}
        else if(year>2 && flg[3]!=1) sorry();
        else if(rdyr[2]<4 && flg[3]==1) inprogress();
        else {oa3[5]=15;
              buy[4]=buysub(oa3[2],oa3[5]);}
        break;
    case'E': case'e':
        buy[5]=buysub(ob1[2],ob1[5]);
        break;
    case'F': case'f':
        if(year==1) notavail();
        else if(year>1 && flg[4]!=1) {mustfund(); printf("MM-III MOD.");}
        else buy[6]=buysub(ob1m[2],ob1m[5]);
        break;
    case'G': case'g':
        if(year==1 && flg[5]!=1) {mustfund(); printf("ALCM.");}
        else if(year>1 && flg[5]!=1) sorry();
        else if(rdyr[3]<3 && flg[5]==1) inprogress();
        else {ob2[5]=15;
              buy[7]=buysub(ob2[2],ob2[5]);}
        break;
    case'H': case'h':
```

```
    if(year==1) notavail();
    else if(year==2 && flg[6]!=1) {mustfund(); printf("MX.");}
    else if(year>2 && flg[6]!=1) sorry();
    else if(rdyr[4]<2 && flg[6]==1) inprogress();
    else {ob3[5]=15;
          buy[8]=buysub(ob3[2],ob3[5]);}
    break;
case'I': case'i':
    if(year==1) notavail();
    else if(year==2 && flg[7]!=1) {mustfund(); printf("FOB.");}
    else if(year>2 && flg[7]!=1) sorry();
    else if(rdyr[5]<3 && flg[7]==1) inprogress();
    else {ob4[5]=15;
          buy[9]=buysub(ob4[2]);}
    break;
case'J': case'j':
    buy[10]=buysub(da1[2],da1[5]);
    break;
case'K': case'k':
    if(year==1) notavail();
    else if(year==2 && flg[8]!=1) {mustfund(); printf("HAWK.");}
    else if(year>2 && flg[8]!=1) sorry();
    else if(rdyr[6]<3 && flg[8]==1) inprogress();
    else {da2[5]=15;
          buy[11]=buysub(da2[2],da2[5]);}
    break;
case'L': case'l':
    if(year==1) notavail();
    else if(year==2 && flg[9]!=1) {mustfund(); printf("PATRIOT.");}
    else if(year>2 && flg[9]!=1) sorry();
    else if(rdyr[7]<2 && flg[9]==1) inprogress();
    else {da3[5]=15;
          buy[12]=buysub(da3[2],da3[5]);}
    break;
case'M': case'm':
    buy[13]=buysub(db1[2],db1[5]);
    break;
```

```
            case'N': case'n':
               if(year==1 && flg[10]!=1) {mustfund(); printf("SPARTAN.");}
               else if(year>1 && flg[10]!=1) sorry();
               else {db2[5]=15;
                     buy[14]=buysub(db2[2],db2[5]);}
               break;
            case'P': case'p':
               if(year==1) notavail();
               else if(year==2 && flg[11]!=1) {mustfund(); printf("LoADS.");}
               else if(year>2 && flg[11]!=1) sorry();
               else if(rdyr[9]<2 && flg[11]==1) inprogress();
               else {db3[5]=15;
                     buy[15]=buysub(db3[2],db3[5]);}
               break;
            case'Q': case'q':
               if(year<3) notavail();
               else if(year==3 && flg[12]!=1) {mustfund(); printf("VADER.");}
               else if(year>3 && flg[12]!=1) sorry();
               else {db4[5]=15;
                     buy[16]=buysub(db4[2],db4[5]);}
               break;
            default:
               printf("YOUR INPUT WAS NOT A VALID CHOICE.");
               break;
            }
         if(timeout==1){done(); trig=0; return;}
         printf("\n\nDO YOU WANT ANY OTHER SYSTEMS? (y/n)  ");
         which();
         }
      return;
}
/*****************************************************
  BUYSUB is a utility subroutine which asks the team
  how many of the selected weapon systems they want
  to buy, checks to be sure that they did not order
  more than the specified purchase limit and tells
  the team how much they spent on the transaction.*/
```

```
buysub(price,max){
   int zbuy;
   getint();
   if(qty>max){
      printf("YOU CANNOT PURCHASE MORE THAN %d OF THESE SYSTEMS PER YEAR.\
         \nI WILL ASSUME YOU WANT THE MAXIMUM.\n",max);
      qty=max;
      }
   zbuy=qty;
   cost=price*qty;
   funds=funds-cost;
   printf("YOU JUST SPENT $%d.\n",cost);
   left();
   return(zbuy);
}
/*************************************************
  WHICH is called by both the RESEARCH and the
  BUY1 subroutines to identify which additional
  systems they wish to buy or R&D.              */

which(){
   yesno();
   switch(sct){
      case 'Y': case'y':
         printf("WHICH SYSTEM? (select letter)\n");
         pur=getans();
         break;
      case 'N': case 'n':
         trig=0;
         break;
      }
}
/*************************************************
  The MODIFY subroutine allows the teams to upgrade
  certain systems in their inventories. The R&D for
  the system upgrades must be performed before the
```

```
  program will allow them to modify.          */

modify()
   {
   newpge();
   if(year==1){
printf("THE ONLY SYSTEM WHICH YOU CAN MOD AT THIS POINT IS THE B-52G.\n\n\n");
      modbuff();
   }
   if(year>=2){
      printf("YOU CAN MODIFY EITHER THE B-52 OR THE MINUTEMAN THIS YEAR\n");
      modbuff();
      printf("\n\n\n\nMOD COSTS FOR THE MM-III ARE %d PER UNIT.",ob1[9]);
      printf("\nDO YOU WISH TO UPGRADE ANY OF YOUR MM-III's? (y/n) ");
      yesno();
      if(sct=='Y'||sct=='y'){
         if(flg[4]!=1){
            sub1();
            return;
         }
         printf("\nYOU HAVE %d UNITS AVAILABLE FOR MODIFICATION",ob1[1]);
         getint();
         if(qty>ob1[5]){
            printf("YOU CANNOT MODIFY MORE THAN %d OF THESE SYSTEMS PER YEAR.\
               \nI WILL ASSUME YOU WANT THE MAXIMUM.\n",ob1[5]);
            qty=ob1[5];
         }
         mmod=qty;
         cost=ob1[9]*qty;
         sub2();
         mmon=qty;
         ob1[1]=ob1[1]-mmod+mmon;
         if(timeout==1) done();
      }
   }
   return;
}
```

```
/*****************************************
  MODBUFF is a utility subroutine used by
  MODIFY for modifications to the offensive
  aircraft system.                        */

modbuff(){
   printf("NOTE: YOU MUST FUND THE R&D BEFORE YOU CAN MODIFY.");
   printf("\n\n\n\n\n\nMOD COSTS FOR THE B52-G ARE %d PER UNIT.",oa1[9]);
   printf("\nDO YOU WISH TO UPGRADE ANY OF YOUR B-52's? (y/n)");
   yesno();
   if(sct=='Y'||sct=='y'){
      if(flg[1]!=1){
         sub1();
         return;
      }
      printf("\nYOU HAVE %d UNITS AVAILABLE FOR MODIFICATION.",oa1[1]);
      getint();
      if(qty>oa1[5]){
         printf("YOU CANNOT MODIFY MORE THAN %d OF THESE SYSTEMS PER YEAR.\
            \nI WILL ASSUME YOU WANT THE MAXIMUM.\n",oa1[5]);
         qty=oa1[5];
      }
      bmod=qty;
      cost=oa1[9]*qty;
      sub2();
      bmon=qty;
      oa1[1]=oa1[1]-bmod+bmon;
      if(timeout==1) done();
   }
}
/*****************************************
  SUB1 and SUB2 are utilities used by
  MODIFY and MODBUFF.

  NOTE:  The reason you see
  so many utility subroutine is that if
  two or more lines of code are used
```

```
  repetitiosly, program space can be saved
  by placing them in a subroutine. Program
  space was a very critical factor during
  most of this program development.    */

sub1(){
   printf("\n\n\n\n\nSORRY, YOU HAVEN'T DONE THE RESEARCH YET.\
      \n\n\n\n\n\n\n\n\nTYPE CARRIAGE RETURN TO GET MENU");
   cont=getchar();
}
sub2(){
   funds=funds-cost;
   printf("YOU JUST SPENT $%d.\n",cost);
   left();
   printf("\n\nYOU MAY CHOOSE HOW MANY OF THE UNITS BEING\
      \nMODIFIED YOU WANT LEFT IN OPERATION WITH THE\
      \nUNMODIFIED UTILITY RATE AND OPERATION COST.");
   getint();
}
/*****************************************
  SCRAP1 is a subroutine which allows the
  teams to deactivate weapon systems which
  they no longer wish to operate.       */

scrap1()
   {
   newpge();
   printf("THE SYSTEMS AVAILABLE FOR YOU TO SCRAP ARE:\n\n");
   printf("                      OPCOST  UTILS\n\n");
   if(oa1[1]>0) printf("A - B-52G BOMBER      %d     %d\n\n",oa1[3],oa1[4]);
   if(oa1m[1]>0)printf("B - UPGRADED B-52G    %d     %d\n\n",oa1m[3],oa1m[4]);
   if(oa2[1]>0) printf("C - FB-111H BOMBER    %d     %d\n\n",oa2[3],oa2[4]);
   if(oa3[1]>0) printf("D - B-1 BOMBER        %d     %d\n\n",oa3[3],oa3[4]);
   if(ob1[1]>0) printf("E - MINUTEMAN III     %d     %d\n\n",ob1[3],ob1[4]);
   if(ob1m[1]>0)printf("F - UPGRADED MM-III   %d     %d\n\n",ob1m[3],ob1m[4]);
   if(ob2[1]>0) printf("G - ALCM              %d     %d\n\n",ob2[3],ob2[4]);
   if(ob3[1]>0) printf("H - MX MOBILE ICBM    %d     %d\n\n",ob3[3],ob3[4]);
```

```
if(ob4[1]>0) printf("I - FOB                %d     %d\n\n",ob4[3],ob4[4]);
if(da1[1]>0) printf("J - CHAPPARAL SAM      %d     %d\n\n",da1[3],da1[4]);
if(da2[1]>0) printf("K - HAWK SAM           %d     %d\n\n",da2[3],da2[4]);
if(da3[1]>0) printf("L - PATRIOT SAM        %d     %d\n\n",da3[3],da3[4]);
if(db1[1]>0) printf("M - SPRINT ABM         %d     %d\n\n",db1[3],db1[4]);
if(db2[1]>0) printf("N - SPARTAN ABM        %d     %d\n\n",db2[3],db2[4]);
if(db3[1]>0) printf("P - LoADS ABM          %d     %d\n\n",db3[3],db3[4]);
if(db4[1]>0) printf("Q - VADER ABM          %d     %d\n\n",db4[3],db4[4]);

printf("SELECT THE SYSTEM YOU WISH TO SCRAP BY TYPING ITS CORRESPONDING");
printf(" LETTER.\n");
pur=getans();
trig=1;
while(trig==1){
   switch(pur){
      case'A':case'a':
         printf("YOU PRESENTLY HAVE %d B-52G's.\n",oa1[1]);
         scrap[1]=dump();
         break;
      case'B':case'b':
         printf("YOU PRESENTLY HAVE %d MODIFIED B-52G's.\n",oa1m[1]);
         scrap[2]=dump();
         break;
      case'C':case'c':
         printf("YOU PRESENTLY HAVE %d FB-111E's.\n",oa2[1]);
         scrap[3]=dump();
         break;
      case'D':case'd':
         printf("YOU PRESENTLY HAVE %d B-1's.\n",oa3[1]);
         scrap[4]=dump();
         break;
      case'E':case'e':
         printf("YOU PRESENTLY HAVE %d MINUTEMAN III's.\n",ob1[1]);
         scrap[5]=dump();
         break;
      case'F':case'f':
         printf("YOU PRESENTLY HAVE %d MODIFIED MM-III's.\n",ob1m[1]);
```

```
      scrap[6]=dump();
      break;
   case'G':case'g':
      printf("YOU PRESENTLY HAVE %d AICM's.\n",ob2[1]);
      scrap[7]=dump();
      break;
   case'H':case'h':
      printf("YOU PRESENTLY HAVE %d MX MOBILE ICBM's.\n",ob3[1]);
      scrap[8]=dump();
      break;
   case'I':case'i':
      printf("YOU PRESENTLY HAVE %d FOB SYSTEMS.\n",ob4[1]);
      scrap[9]=dump();
      break;
   case'J':case'j':
      printf("YOU PRESENTLY HAVE %d CHAPPARALS.\n",da1[1]);
      scrap[10]=dump();
      break;
   case'K':case'k':
      printf("YOU PRESENTLY HAVE %d IMPROVED HAWKS.\n",da2[1]);
      scrap[11]=dump();
      break;
   case'L':case'l':
      printf("YOU PRESENTLY HAVE %d PATRIOTS.\n",da3[1]);
      scrap[12]=dump();
      break;
   case'M':case'm':
      printf("YOU PRESENTLY HAVE %d SPRINTS.\n",db1[1]);
      scrap[13]=dump();
      break;
   case'N':case'n':
      printf("YOU PRESENTLY HAVE %d SPARTANS.\n",db2[1]);
      scrap[14]=dump();
      break;
   case'P':case'p':
      printf("YOU PRESENTLY HAVE %d LoADS ABM'S.\n",db3[1]);
      scrap[15]=dump();
```

```
                break;
            case'Q':case'q':
                printf("YOU PRESENTLY HAVE %d VADERS.\n",dt4[1]);
                scrap[16]=dump();
                break;
            default:
                printf("YOUR INPUT WAS NOT A VALID CHOICE.");
                break;
            }
        printf("\n\nDO YOU WANT TO SCRAP ANY OTHER SYSTEMS? (y/n)  ");
        whicn();
        }
    return;
}
/*************************************************
  DUMP is a utility subroutine called by SCRAP1
  to determine how many units are to be deacti-
  vated and it advises the team that their
  request has been accepted.                  */

dump(){
    int zscrap;
    printf("HOW MANY DO YOU WISH TO SCRAP?\n");
    for(n=0;((abuf[n]=getchar())!='\n')&&(n<10);n++);
    abuf[n]='\0';
    qty=atoi(abuf);
    zscrap=qty;
    printf("NEXT YEAR'S INVENTORY WILL BE %d LESS.",zscrap);
    return(zscrap);
}
/*************************************************
  The INTEL subroutine allows the team to select
  intelligence data they went to purchase about
  the opponent's force structure or R&D activities.
  It sets flags based on these requests which are
  acted upon in the DONE subroutine.          */
```

```
intel(){
   if(year<=5){
      stop=0;
      newpge();
      printf("WHAT TYPE OF INTELLIGENCE DO YOU WISH TO OBTAIN?\
      \n\nYOUR CHOICES ARE:\
      \n\n         1 - CURRENT FORCE STRUCTURE OF OFFENSIVE FORCES\
      \n\n         2 - CURRENT FORCE STRUCTURE OF DEFENSIVE FORCES\
      \n\n         3 - CURRENT R&D PROGRAMS FOR OFFENSIVE FORCES\
      \n\n         4 - CURRENT R&D PROGRAMS FOR DEFENSIVE FORCES\
      \n\n         5 - OFFENSIVE FORCE COUNTERINTELLIGENCE\
      \n\n         6 - DEFENSIVE FORCE COUNTERINTELLIGENCE\
      \n\n         7 - OFFENSIVE R&D COUNTERINTELLIGENCE\
      \n\n         8 - OFFENSIVE R&D COUNTERINTELLIGENCE\
      \n\n         9 - NO FURTHER INTELLIGENCE INFO DESIRED\
      \n\n         0 - CANCEL ALL INTELLIGENCE REQUESTS ENTERED THIS YEAR\
      \n\n\n\n\n\nPLEASE SELECT 1-6 \n\n\n");
      while(stop==0){
         pur=getans();
         switch(pur){
            case '1':
               intflag[1]=1;
               intmsg();
               break;
            case '2':
               intflag[2]=1;
               intmsg();
               break;
            case '3':
               intflag[3]=1;
               intmsg();
               break;
            case '4':
               intflag[4]=1;
               intmsg();
               break;
            case '5':
```

```
                intflag[5]=1;
                intmsg1();
                break;
            case '6':
                intflag[6]=1;
                intmsg1();
                break;
            case '7':
                intflag[7]=1;
                intmsg1();
                break;
            case '8':
                intflag[8]=1;
                intmsg1();
                break;
            case '0':
                for(i=1;i<5;i++){
                    if(intflag[i]==1) funds=funds+100;
                    intflag[i]=0;
                }
                for(i=5;i<9;i++){
                    if(intflag[i]==1) funds=funds+50;
                    intflag[i]=0;
                }
printf("\nYOUR PREVIOUS DEDUCTIONS FOR INTEL THIS YEAR HAVE BEEN RESTORED.\
 \nDO YOU WANT ANY MORE INTEL? (choose 1-0)\n");
                break;
            default:
                stop=1;
                break;
        }
    }
}
else
    printf("\nYOUR INTELLIGENCE GATHERING CAPABILITIES HAVE BEEN DESTROYED.\
 \nYOU CANNOT OBTAIN ANY MORE REPORTS FOR THE DURATION OF THE\
 \nGAME.");
```

```
    return;
}
/*************************************************
  INTMSG and INTMSG1 are utility subroutines called
  by INTEL to advise the team how much they spent
  with each transaction. Intelligence costs $100
  per request and counterintelligence costs $50.*/

intmsg(){
    funds=funds-100;
    printf("\nYOU HAVE JUST SPENT $100.");
    left();
    if(timeout==1){done(); return;}
    printf("DO YOU WANT ANY MORE INTEL? (choose 1-0)\n");
}

intmsg1(){
    funds=funds-50;
    printf("\nYOU HAVE JUST SPENT $50.");
    left();
    if(timeout==1){done(); return;}
    printf("DO YOU WANT ANY MORE INTEL? (choose 1-0)\n");
}
/*************************************************
  The ANALYZE subroutine is used to call the
  analysis program from outside this main program
  structure.  The ANALYZE program was implemented
  as an overlay to conserve program space in
  BLUSIDE. This subroutine also advises the team
  of their expenditures for analysis.          */

analyze()
    {
    int fk;
    if(year<=4){
       if((fk=fork())==0){
          execl("/usr/tempoblu/analyze",&year,0);
```

```
            exit();
         }
         wait(&fk);
         upay= open("/usr/tempoblu/analpay",2);
         read(upay,anal,3);
         close(upay);
         acost=atoi(anal);
         acostt=acost*25;
         funds=funds-acostt;
         newpge();
printf("\nYOUR ANALYSIS OF %d PAIRS OF SYSTEMS COST YOU $%d.\n",acost,acostt);
         left();
      }
      else
         printf("\n\n\n\n\n\n\n\nYOUR ANALYSTS HAVE QUIT AND GONE TO WORK FOR\
            \nPRIVATE INDUSTRY. SORRY, NO MORE ANALYSIS BRANCH.");
      printf("\n\n\n\n\n\n\nHIT RETURN FOR MAIN MENU.");
      cont=getchar();
      return;
}
/*************************************************
  The REINIT subroutine calls the RESET program
  from outside this main program structure. This
  is the tutorial which advises the team how to
  recover from certain entry errors.  The use of
  an overlay structure was implemented to save
  program space in BLUSIDE.                    */

reinit()
   {
   int fk;
   if((fk=fork())==0){
      execl("/usr/tempoblu/reset",0);
      exit();}
   wait(&fk);
   return;
}
```

```
/*****************************************************************
** This is BLUSID1, a continuation of BLUSIDE. This was        **
** an implementation revealed by BBN which allows for          **
** more available program space.  By segmenting the program*
** into two or more pieces and compiling them separately, **
** and then linking them together in the object code, a        **
** much larger program can be accommodated. The compiler       **
** is invoked using the following command string:              **
**                    cc -i bluside.c blusid1.c                **
** All global variable which are used by both program          **
** segments must be declared in both segments which accounts
** for this apparent redeclaration of variables.               **
*****************************************************************/
float x;
double fmod();
int atoi(),rand(),srand(),hrstrt,minstrt,quit;
int tvec[2],time(),i,t,hleft,mleft,tdisp,timebst;
int *localtime(),*locl,clock[3];
int m,p,q,mode;
int oa1[11];
int oa1m[11];
int oa2[11];
int ob1[11];
int ob2[11];
int da1[11];
int db1[11];
int db2[11];
int oa3[10],ob1m[10],ob3[10];
int ob4[10],da2[10],da3[10];
int cost,blufb,loss,byrrep,app1,app2,pw,pw1;
int db3[10],db4[10];
int year,n,funds,bucks,qty,trig,intel,theend,tempflg[15];
int z,flg[15],stop,intflag[10],rdyr[11],tcost,tutil,apprp,iwar;
int offa,offb,defa,defb,ops1,ops2,ops3,ops4,opcost,spent,deficit;
```

```
int buy[17],scrap[17],bmod,mmod,bmon,mmon,send,timeout,warrep,warmsg;
char cont,sel,sct,war[5],abuf[10],pur,clear,news[20],finale[178];
char str[5];
char sstr[3];
/*****************************************************************
  DONE is the most complex subroutine in the entire TEMPO
  program.  It performs all the year-end accounting and
  data transferring functions between the two teams and
  the umpire.                                                  */

done()
   {
   int fk;
   newpge();
   if(timeout==1) printf("NOTE FOR BIG SPENDERS: YOU WILL NOT BE BLOCKED\
                         \nFROM RETURNING TO THE MAIN MENU,HOWEVER, I WOULD\
                         \nSTRONGIY RECOMMEND THAT YOU NOT SPEND MUCH MORE.\
                         \nTHAT'S NO WAY TO WIN A WAR.\n\n");

   printf("IF YOU ARE CERTAIN THAT YOU HAVE MADE ALL THE ENTRIES\
         \nTHAT YOU WANT FOR THIS YEAR, CONFIRM WITH A 'YES'.\
         \nOTHERWISE, 'NO' WILL RETURN YOU TO THE MENU. (y/n)   \n");
   yesno();
   switch(sct){
      case'Y':case'y':
         timeout=0;
         utility();
         costops();
         funds=funds-opcost;
         spent=apprp-funds;
         if(year<5||year==8) early();
         if(year<5||year==8) printf("%c",clear);
         else
            newpge();
         printf("\n\nHERE IS A BRIEF SUMMARY OF YEAR %d .\
                \n\nYOUR TOTAL OFFENSIVE A UTILS WERE %d.\
                \nYOUR TOTAL OFFENSIVE B UTILS WERE %d.",year,offa,offb);
```

```
        printf("\nYOUR TOTAL DEFENSIVE A UTILS WERE %d.\
                \nYOUR TOTAL DEFENSIVE B UTILS WERE %d.",defa,defb);
printf("\n\nTHE TOTAL OPERATIONS COSTS FOR YOUR SYSTEMS WERE %d.",opcost);
        printf("\n\nYOUR TOTAL EXPENDITURES FOR THIS YEAR WERE %d.",spent);
        if(spent>apprp){
            deficit=spent-apprp;
            printf("\n\nUNFORTUNATELY, YOU DIDN'T PLAN AS WELL AS YOU\
                    \nSHOULD HAVE. YOU HAD A DEFICIT OF $%d FOR THE YEAR.\
                    \nNEXT YEAR'S BUDGET WILL BE REDUCED BY TWICE\
                    \nTHIS AMOUNT.",deficit);
            yearend();
            umpire();
            year=year+1;
            database();
            if(year<=5||year==9) timechk();
            apprp=apprp-(2*deficit);
            funds=apprp;
            }
        else{
            yearend();
            umpire();
            year=year+1;
            database();
            if(year<=5||year==9) timechk();
            }
        byrrep=open("/usr/umpire/byrrep",2);
        read(byrrep,news,20);
        printf("\n\n%s\n",news);
        close(byrrep);

        oa1[1]=oa1[1]+buy[1]-scrap[1];          oa1m[1]=oa1m[1]+buy[2]-scrap[2];
        oa2[1]=oa2[1]+buy[3]-scrap[3];          oa3[1]=oa3[1]+buy[4]-scrap[4];
        ob1[1]=ob1[1]+buy[5]-scrap[5];          ob1m[1]=ob1m[1]+buy[6]-scrap[6];
        ob2[1]=ob2[1]+buy[7]-scrap[7];          ob3[1]=ob3[1]+buy[8]-scrap[8];
        ob4[1]=ob4[1]+buy[9]-scrap[9];          da1[1]=da1[1]+buy[10]-scrap[10];
        da2[1]=da2[1]+buy[11]-scrap[11];        da3[1]=da3[1]+buy[12]-scrap[12];
        db1[1]=db1[1]+buy[13]-scrap[13];        db2[1]=db2[1]+buy[14]-scrap[14];
```

```
        db3[1]=db3[1]+buy[15]-scrap[15];      db4[1]=db4[1]+buy[16]-scrap[16];

        oa1[1]=oa1[1]-bmon;                   oa1m[1]=oa1m[1]+bmod;
        ob1[1]=ob1[1]-mmon;                   ob1m[1]=ob1m[1]+mmod;

        for(i=1;i<=16;i++) buy[i]=0;
        for(i=1;i<=16;i++) scrap[i]=0;
        bmod=0;
        mmod=0;

        offint();
        defint();
        ordint();
        drdint();
        cntrint();
        for(i=1;i<14;i++) tempflg[i]=0;
        printf("\n\n\n\n
**********************************************************************************\
*\n");
        if(mode==1){
              if(year<=5){
                printf("\n
                  YOU HAVE A FIVE MINUTE BREAK TO REGROUP.\n");
/*              sleep(295);                                          */
              }
        }
        if(mode==2){
           if(year==2){printf("\n
              YOU HAVE FIFTEEN MINUTES TO RELAX AND DISCUSS THE\n
              FIRST YEAR PRIOR TO THE BEGINNING OF YEAR 2 PLANNING.");
              sleep(890);
           }
           if(year>2&&year<=5){
              printf("\n\n\n\n
                      YOU HAVE A TEN MINUTE BREAK.");
              sleep(590);
           }
```

```
            }
            if(year<9){
               wartime();
               printf("\n\n\n\nPRESS RETURN FOR NEXT YEAR.");
               cont=getchar();
               if(year<=5||year==9){
                  hrstrt=clock[2];
                  minstrt=clock[1];
               }
               yrbegin();
               intrep();
            }
            if(year==9){
               theend = open("/usr/umpire/lastrep",2);
               read(theend,finale,178);
               close(theend);
               printf("%c",clear);
               printf("%s",finale);
               quit=1;
            }
            break;
         case'N':case'n':
            break;
         }
   return;
}
/*****************************************************************
  The UTILITY subroutine discounts the utility of each
  weapon system accrding to a scheme described in the
  User's Manual.                                              */

utility(){
   int d[17];
   offa=oa1[1]*oa1[4];
   d[1]=discount(offa);
   offa=oa1m[1]*oa1m[4];
   d[2]=discount(offa);
```

```
    offa=ca2[1]*ca2[4];
    d[3]=discount(offa);
    offa=ca3[1]*oa3[4];
    d[4]=discount(offa);
    offb=ob1[1]*ob1[4];
    d[5]=discount(offb);
    offb=ob1m[1]*ob1m[4];
    d[6]=discount(offb);
    offb=ob2[1]*ob2[4];
    d[7]=discount(offb);
    offb=ob3[1]*ob3[4];
    d[8]=discount(offb);
    offb=ob4[1]*ob4[4];
    d[9]=discount(offb);
    defa=da1[1]*da1[4];
    d[10]=discount(defa);
    defa=da2[1]*da2[4];
    d[11]=discount(defa);
    defa=da3[1]*da3[4];
    d[12]=discount(defa);
    defb=db1[1]*db1[4];
    d[13]=discount(defb);
    defb=db2[1]*db2[4];
    d[14]=discount(defb);
    defb=db3[1]*db3[4];
    d[15]=discount(defb);
    defb=db4[1]*db4[4];
    d[16]=discount(defb);
    offa=0; offb=0; defa=0; defb=0;

    for(i=1;i<=3;i++) offa=offa+d[i];
    for(i=4;i<=9;i++) offb=offb+d[i];
    for(i=10;i<=12;i++) defa=defa+d[i];
    for(i=13;i<=16;i++) defb=defb+d[i];
}
/**********************************************************************
  DISCOUNT is a utility subroutine used by UTILITY to
```

```
  perform the arithmetic necessary in the discounting
  process.                                                    */

discount(gross){
   float disct;
   int disc;
   if(gross<=2000) disct=gross;
   if(gross>2000 && gross<=3000) disct=2000.+(.9*(gross-2000));
   if(gross>3000 && gross<=4000) disct=2900.+(.8*(gross-3000));
   if(gross>4000 && gross<=5000) disct=3700.+(.7*(gross-4000));
   if(gross>5000 && gross<=6000) disct=4400.+(.6*(gross-5000));
   if(gross>6000 && gross<=7000) disct=5000.+(.5*(gross-6000));
   if(gross>7000 && gross<=8000) disct=5500.+(.4*(gross-7000));
   if(gross>8000 && gross<=9000) disct=5900.+(.3*(gross-8000));
   if(gross>9000 && gross<=10000) disct=6200.+(.2*(gross-9000));
   if(gross>10000 && gross<=11000) disct=6400.+(.1*(gross-10000));
   if(gross>11000) disct=6500.;
   disc=disct;
   return(disc);
}
/*************************************************************
  The EARLY subroutine is called by DONE if the team
  completes their planning in less than the allotted time.
  This routine will hold both sides up until the prescribed
  time for proceeding to the year-end accounting to insure
  that all data transfers are timed correctly.                */

early(){
   int tprev;
   if(tdisp>0){
      printf("%c",clear);
      printf("\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n
  \n                      YOU FINISHED A LITTLE EARLY THIS YEAR.\
  \n\n                      STAND BY UNTIL THE ALLOCATED TIME FOR\
  \n\n                      PLANNING IS COMPLETE.\
  \n\n\n\n                        OBSERVE THAT YOU GAIN NOTHING BY FINISHING\
  \n\n                      AHEAD OF TIME.  TAKE ADVANTAGE OF THE TIME\
```

```
\n\n                          YOU ARE GIVEN.");
    printf("\n\n\n\n\n\n");
    tprev=100;
    while(tdisp>0){
        time(tvec);
        locl = localtime(tvec);
        for(i=0;i<3;i++) clock[i]=locl[i];
        if(clock[1]<minstrt){
           clock[2]=clock[2]-1;
           clock[1]=clock[1]+60;
        }
        hleft=clock[2]-hrstrt;
        mleft=clock[1]-minstrt;
        if(mode==1){
           if(year==1) t=35;
           if(year==2) t=20;
           if(year==3) t=15;
           if(year==4) t=10;
           if(year==5) t=20;
        }
        else{
           if(year==1) t=50;
           if(year==2) t=45;
           if(year==3) t=35;
           if(year==4) t=30;
           if(year==5) t=35;
        }
        tdisp = t-mleft;
        if(tdisp!=tprev)
        printf("\n%d MIN LEFT",tdisp);
        tprev = tdisp;
    }
  }
}
/*****************************************************************
  TIMECHK is called by DONE to determine if the budget was
  submitted late.  If so, the team is penalized in the
```

```
  following year's budget.                                    */

timechk(){
   if(tdisp<0){
      timebst =  -tdisp;
      if(timebst<=5)
         apprp=apprp-(timebst*50);
      if(timebst>5)
         apprp=apprp-250-((timebst-5)*100);
      printf("\n\nYOUR PLAN FOR THIS YEAR WAS SUBMITTED %d MINUTES\
         \nLATE.  YOUR BUDGET FOR NEXT YEAR WILL BE REDUCED BY\
         \n$50 FOR EACH MINUTE LATE FOR THE FIRST FIVE MINUTES\
         \nAND $100 PER MINUTE FOR EACH MINUTE OVER FIVE.\n",timebst);
   }
   return;
}
/*************************************************************
  COSTOPS is a utility subroutine called by DONE and by
  BUDGET to calculate the cost of operations for the current
  active force.                                               */

costops(){
   ops1=oa1[1]*oa1[3]+oa1m[1]*oa1m[3]+oa2[1]*oa2[3]+oa3[1]*oa3[3];
   ops2=ob1[1]*ob1[3]+ob1m[1]*ob1m[3]+ob2[1]*ob2[3]+ob3[1]*ob3[3];
   ops2=ops2+ob4[1]*ob4[3];
   ops3=da1[1]*da1[3]+da2[1]*da2[3]+da3[1]*da3[3];
   ops4=db1[1]*db1[3]+db2[1]*db2[3]+db3[1]*db3[3]+db4[1]*db4[3];
   opcost=ops1+ops2+ops3+ops4;
}
/*************************************************************
  WARTIME is a subroutine called by done to determine if a
  war has occurred during the current year. The war/no war
  flag is read from the umpire file WAR. If there was a
  war the file BLUWAR1 is read to determine the outcome.
  This message is displayed at the team's terminal and
  then the file BLUFDBK is read to determine the budget
  cost of the war for this team.                              */
```

```
wartime(){
   char notes[650];
   warrep = open("/usr/umpire/war",2);
   read(warrep,war,5);
   close(warrep);
   iwar = atoi(war);
   if(iwar==1)  {
      warmsg = open("/usr/tempoblu/bluwar1",2);
      read(warmsg,notes,650);
      close(warmsg);
      printf("%c",clear);
      printf("%s",notes);
      blufb = open("/usr/tempoblu/blufdbk",2);
      read(blufb,str,5);
      close(blufb);
      loss = atoi(str);
      apprp=apprp-loss;
      funds=apprp;
      printf("\n\n\n\n<CR> to continue");
      ccnt=getchar();
   }
   return;
}
/*****************************************************************
  INTREP is a subroutine called by DONE to read intelligence
  data from REDSIDE files.  Depending on which types of
  intelligence flags were set in the subroutine INTEL,
  various files are read and output to the BLUSIDE terminal*/

intrep(){
   int intfil1,intfil2,intfil3,intfil4,ciof,cidf,cirdo,cirdd,ci;
   char reprt1[215],reprt2[215],reprt3[215],reprt4[215],cistr[3];
   if(year<=5){
      for(i=1;i<9;i++) if(intflag[i]==1)
         printf("%c",clear);
      if(intflag[5]==1||intflag[6]==1||intflag[7]==1||intflag[8]==1){
```

```
    ci= open("/usr/tempored/cntrint",2);
    read(ci,cistr,3);
    ciof=atoi(cistr);
    read(ci,cistr,3);
    cidf=atoi(cistr);
    read(ci,cistr,3);
    cirdo=atoi(cistr);
    read(ci,cistr,3);
    cirdd=atoi(cistr);
    close(ci);
}
if(intflag[1]==1){
    intfil1 = open("/usr/tempored/redoff",2);
    read(intfil1,reprt1,214);
    printf("%s",reprt1);
    close(intfil1);
}
if(intflag[2]==1){
    intfil2 = open("/usr/tempored/reddef",2);
    read(intfil2,reprt2,210);
    printf("%s",reprt2);
    close(intfil2);
}
if(intflag[1]==1&&intflag[2]==1&&intflag[3]==1&&intflag[4]==1){
    printf("\n\nTO GET YOUR OTHER REPORTS HIT RETURN.\
       \nNOTE: YOU WILL NOT SEE THESE REPORTS AGAIN\
       \nSO YOU WOULD BE WELL ADVISED TO RECORD THE\
       \nDATA YOU NEED.");
    cont=getchar();
    newpge();
}
if(intflag[3]==1){
    intfil3 = open("/usr/tempored/redrdo",2);
    read(intfil3,reprt3,215);
    printf("%s",reprt3);
    close(intfil3);
}
```

```
if(intflag[4]==1){
   intfil4 = open("/usr/tempored/redrdd",2);
   read(intfil4,reprt4,215);
   printf("%s",reprt4);
   close(intfil4);
}
if(intflag[5]==1){
   if(ciof==0)
      printf("\nNO INFORMATION ON YOUR OFFENSIVE FORCE STATUS\
         \nWAS COMPROMISED THIS YEAR.\n");
   else
      printf("\nENEMY AGENTS HAVE DISCOVERED AND REPORTED YOUR\
         \nCOMPLETE OFFENSIVE FORCE STATUS FOR THIS YEAR.\n");
}
if(intflag[6]==1){
   if(cidf==0)
   printf("\nTHE ENEMY IS NOT AWARE OF YOUR DEFENSIVE FORCE\
         \nSTATUS FOR THIS YEAR.\n");
   else
      printf("\nENEMY AGENTS HAVE OBTAINED DOCUMENTS COMPLETELY\
         \nDETAILING YOUR DEFENSIVE FORCE STATUS THIS YEAR.\n");
}
if(intflag[7]==1){
   if(cirdo==0)
      printf("\nTHE ENEMY'S ATTEMPTS TO INFILTRATE THE DEVELOPER'S\
       \nFACILITIES WERE UNSUCCESSFUL. HE KNOWS NOTHING\
         \nOF THIS YEAR'S OFFENSIVE R&D PROGRAMS.\n");
      else
         printf("\nYOUR OFFENSIVE SYSTEMS R&D ARE NO SECRET THIS\
            \nYEAR. THE OTHER SIDE KNOWS ALL.\n");
}
if(intflag[8]==1){
   if(cirdd==0)
      printf("\nYOUR SECURITY ON DEFENSIVE SYSTEMS R&D HELD UP\
         \nTHIS YEAR. THE ENEMY DOES NOT HAVE ANY SUCH INFO.\n");
   else
      printf("\nNUMEROUS LEAKS IN HIGH PLACES HAVE GIVEN THE\
```

```
                \nENEMY YOUR COMPLETE DEFENSIVE R&D STATUS FOR THE YEAR.\n");
        }
        if(intflag[1]==1&&intflag[2]==1&&intflag[3]==1&&intflag[4]==1){
           printf("\n\n\nPRESS RETURN TO CONTINUE.");
           cont=getchar();
        }
        if(intflag[1]!=1||intflag[2]!=1||intflag[3]!=1||intflag[4]!=1){
           if(intflag[1]==1||intflag[2]==1||intflag[3]==1||intflag[4]==1)
              printf("\n\nNOTE: YOU WILL NOT SEE THESE REPORTS AGAIN\
                      \nSO YOU WOULD BE WELL ADVISED TO RECORD THE\
                      \nDATA YOU NEED.\
                      \n\nPRESS RETURN TO CONTINUE.");
           cont=getchar();
        }
        for(i=1;i<9;i++) intflag[i]=0;
     }
}
/*****************************************************************
  The UMPIRE subroutine calls the external program UMPIRE
  at the end of the year to record the total utility for
  each of the weapon system categories.  These totals are
  compiled in a table along with the REDSIDE totals and a
  winner for the year is determined.  A cumulative total
  is also kept so an overall game winner can be fetermined.
  This routine is the only difference between the BLUSIDE
  and REDSIDE program structures.  In REDSIDE the utility
  totals are written into a file for UMPIRE to read whereas
  in BLUSIDE, the values are carried through the fork to
  UMPIRE.  Notice that the integer utility values must be
  converted to character values for the transfer.  All
  I/O in UNIX is in character format.                    */

umpire(){
   int fk;
   char yearc[5],offac[5],offbc[5],defac[5],defbc[5],randc[5];
   x = year;
   itoa(x);
```

```
    for(i=0;i<5;i++) yearc[i]=str[i];
    x = offa;
    itoa(x);
    for(i=0;i<5;i++) offac[i]=str[i];
    x = offb;
    itoa(x);
    for(i=0;i<5;i++) offbc[i]=str[i];
    x = defa;
    itoa(x);
    for(i=0;i<5;i++) defac[i]=str[i];
    x = defb;
    itoa(x);
    for(i=0;i<5;i++) defbc[i]=str[i];
    x = rand();
    itoa(x);
    for(i=0;i<5;i++) randc[i]=str[i];

    if((fk=fork())==0){
       execl("/usr/umpire/umpire",yearc,offac,offbc,defac,defbc,randc,0);
       exit();
    }
    wait(&fk);
    return;
}
/*************************************************************************
  YRBEGIN is a subroutine called by DONE just before the
  team begins the next year's planning cycle.  This routine
  provides introductory remarks and data prior to the
  beginning of the year.                                          */

yrbegin(){
   newpge();
   if(year<5){
      printf("\n\n\n\n\n
                  THIS IS YEAR %d OF THE GAME.\n\n
                  YOUR BUDGET FOR THIS YEAR IS $%d.\n
                          BUDGET ESTIMATE FOR NEXT YEAR IS $%d.\n
```

```
                        BUDGET ESTIMATE FOR THE FOLLOWING YEAR IS $%d.\n\n
                    THE PROBABILITY OF WAR FOR THIS YEAR IS 0.%d\n
                        INTELLIGENCE ESTIMATES THE PROBABILITY OF\n
                        WAR FOR NEXT YEAR TO BE 0.%d\n\n
                    IF YOU ORDERED INTELLIGENCE REPORTS, THEY WILL\n
                        APPEAR NEXT ON THE SCREEN.\n\n\n
                    FOR MORE DETAIL ON THIS YEAR'S FORCE STATUS AND\n
                        PROGRAMS SELECT THE DESIRED ENTRY FROM THE\n
                        FOLLOWING MENU.\n\n\n\n
                    PRESS RETURN TO CONTINUE.\n",year,apprp,app1,app2,pw,pw1);
        cont=getchar();
    }
    if(year==5){
        printf("\n\n\n\n\n
                    THIS IS YEAR %d OF THE GAME.\n\n
                    YOUR BUDGET FOR THIS YEAR IS $%d.\n
                        BUDGET FOR EACH OF THE NEXT THREE YEARS IS $%d.\n\n
                    THE PROBABILITY OF WAR FOR THIS YEAR IS 0.%d\n
                        INTELLIGENCE ESTIMATES THE PROBABILITY OF\n
                        WAR FOR NEXT THREE YEARS TO BE 0.%d\n\n\n
                    FOR MORE DETAIL ON THIS YEAR'S FORCE STATUS AND\n
                        PROGRAMS, SELECT THE DESIRED ENTRY FROM THE\n
                        FOLLOWING MENU.\n\n\n\n
                    PRESS RETURN TO CONTINUE.\n",year,appri,apprp,pw,pw1);
        cont=getchar();
    }
    return;
}
/*****************************************************************
  NEWPGE is a utility subroutine called through the program
  to clear the screen and update the countdown clock.     */

newpge(){
    printf("%c",clear);
    gclock();
    return;
}
```

```
/*************************************************************
  GETANS is a utility subroutine which accepts a keyboard
  input and rejects any characters beyond the first one
  entered.                                                  */

getans(){
   char buf[20],k;
   int l;
   for(l=0;(k=getchar())!='\n';l++) buf[l]=k;
   k=buf[0];
   return(k);
}
/*************************************************************
  GETINT is a utility subroutine which accepts a character
  keyboard input and converts it to an integer value.      */

getint(){
   printf("\nHOW MANY DO YOU WANT?\n");
   for(n=0;((abuf[n]=getchar())!='\n')&&(n<10);n++);
   abuf[n]='\0';
   qty=atoi(abuf);
   return;
}
/*************************************************************
  YESNO is a utility subroutine used to accept yes or no
  inputs from the keyboard.  It rejects any other entry.  */

yesno(){
   sct=getans();
   while(sct!='y'&&sct!='Y'&&sct!='n'&&sct!='N'){
      printf("Y OR N PLEASE!");
   sct=getans();
   }
}
/*************************************************************
  The DATABASE subroutine is used to update all system
  parameters regarding the available weapon systems each
```

```
   year. Also in the yearly database updates are the budget
   and projected budget figures and the current and projected
   probabilities of war.                                        */

database()
   {
        /*      first parameter: current inventory
                second parameter:acquisition cost per unit
                third parameter: operating cost per unit per year
                fourth parameter:utils per unit
                fifth parameter: purchase limit per year
                sixth parameter: first year R&D cost
                seventh parameter:second year R&D cost
                eighth parameter:third year R&D cost
                ninth parameter: modification cost per unit
                tenth parameter: number of years until available    */

   switch(year){
      case 1:    funds=7500;     pw=25;      app1=6800;
                 apprp=7500;     pw1=37;     app2=7100;

        oa2[5]=0;       ob1m[5]=0;      ob2[5]=0;       ob3[5]=0;
        oa3[5]=0;       da2[5]=0;       da3[5]=0;       db2[5]=0;
        db3[5]=0;       ob4[5]=0;       db4[5]=0;

        break;

      case 2:
         funds=8000;
         apprp=8000;        pw1=25;
         app1=8500;
         app2=9500;
         if(iwar==1) pw=05;
         else pw=15;

        oa2[2]=150;     ob1m[2]=100;    ob2[2]=120;
        oa2[3]= 50;     ob1m[3]= 60;    ob2[3]=100;
```

```
    oa2[4]=125;     ob1m[4]= 70;    ob2[4]=180;
    oa2[6]=000;     ob1m[6]=100;    cb2[6]=000;
    oa2[7]=900;     ob1m[7]=000;    ob2[7]=800;
    oa2[8]=900;     ob1m[8]=000;    cb2[8]=400;
    oa2[9]=000;     ob1m[9]=000;    ob2[9]=000;
    oa2[10]= 2;     ob1m[10]= 1;    ob2[10]= 2;

    cb3[2]=150;     da2[2]=160;     db2[2]= 90;
    ob3[3]=100;     da2[3]=140;     db2[3]= 60;
    ob3[4]=280;     da2[4]=300;     db2[4]=110;
    ob3[6]=300;     da2[6]=300;     db2[6]=000;
    ob3[7]=600;     da2[7]=1000;    db2[7]=400;
    cb3[8]=800;     da2[8]=1000;    db2[8]=000;
    ob3[9]=000;     da2[9]=000;     db2[9]=000;
    ob3[10]= 3;     da2[10]= 3;     db2[10]= 1;

    db3[2]=100;
    db3[3]= 50;
    db3[4]=200;
    db3[6]=1000;
    db3[7]=000;
    db3[8]=000;
    db3[9]=000;
    db3[10]=1 ;

    break;

case 3:
    funds=8400;
    apprp=8400;     db3[6]=0;
    app1=9000;
    app2=8500;      pw1=50;
    if(iwar==1) pw=10;
    else pw=35;

    oa3[2]=200;
    ca3[3]= 80;
```

```
    oa3[4]=200;
    oa3[6]=800;
    oa3[7]=800;
    oa3[8]=800;
    oa3[9]=000;
    oa3[10]= 3;

    ob4[2]=100;
    ob4[3]= 80;
    ob4[4]=160;
    ob4[6]=800;
    ob4[7]=600;
    ob4[8]=000;
    ob4[9]=000;
    ob4[10]= 2;

    oa2[2]=200;      ob2[2]=130;      ob3[2]=180;
    oa2[3]=50 ;      ob2[3]=100;      ob3[3]=110;
    oa2[4]=100;      ob2[4]=200;      ob3[4]=300;
    oa2[6]=000;      ob2[6]=000;      ob3[6]=000;
    oa2[7]=900;      ob2[7]=800;      ob3[7]=500;
    oa2[8]=700;      ob2[8]=400;      ob3[8]=500;
    oa2[9]=000;      ob2[9]=000;      ob3[9]=000;
    oa2[10]=1 ;      ob2[10]=1 ;      ob3[10]=2 ;

    da2[2]=160;      da3[2]=200;
    da2[3]=140;      da3[3]=60 ;
    da2[4]=350;      da3[4]=200;
    da2[6]=000;      da3[6]=1000;
    da2[7]=800;      da3[7]=1800;
    da2[8]=900;      da3[8]=000;
    da2[9]=000;      da3[9]=000;
    da2[10]=2 ;      da3[10]=2 ;

    break;

case 4:
```

```
funds=8700;
apprp=8700;
app1=9000;
app2=9100;      pw1=65;
if(iwar==1) pw=15;
else pw=45;

oa3[2]=225;
oa3[3]=80 ;
oa3[4]=240;
oa3[6]=000;
oa3[7]=1100;
oa3[8]=900;
oa3[9]=000;
oa3[10]=2 ;

ob4[2]=120;
ob4[3]=70 ;
ob4[4]=150;
ob4[6]=000;
ob4[7]=800;
ob4[8]=000;
ob4[9]=000;
ob4[10]=1 ;


ob3[2]=180;      da2[2]=150;      da3[2]=150;
ob3[3]=120;      da2[3]=120;      da3[3]=50 ;
ob3[4]=330;      da2[4]=400;      da3[4]=250;
ob3[6]=000;      da2[6]=000;      da3[6]=000;
ob3[7]=500;      da2[7]=800;      da3[7]=1500;
ob3[8]=300;      da2[8]=800;      da3[8]=00;
ob3[9]=000;      da2[9]=000;      da3[9]=000;
ob3[10]=1 ;      da2[10]=1 ;      da3[10]=1 ;

break;
```

```
        case 5:
           funds=9000;
           apprp=9000;
           if(iwar==1) pw=25;
           else pw=65;
           pw1=25;

           oa3[2]=250;
           oa3[3]=80;
           oa3[4]=280;
           oa3[6]=000;
           oa3[7]=1100;
           oa3[8]=1000;
           oa3[9]=000;
           oa3[10]=1 ;

           break;
        case 6:   funds=9000;
                  apprp=9000;
                  break;
        case 7:   funds=9000;
                  apprp=9000;
                  break;
        default:  funds=9000;
                  apprp=9000;
                  break;

        }
    return;
}
/*****************************************************************
  ITOA is a utility subroutine which converts integer values
  into character strings so they can be written into files
  and carried into other programs.                              */

itoa(){
   int j,k;
```

```
    float y;
    char rstr[5];
    k=0;
    while(k<=4){
       y = fmod(x,10.);
       i = y;
       rstr[k] = i + '0';
       x = (x-y)/10.;
       k = k+1;
    }
    for(j=0;j<=4;j++) str[j]=rstr[4-j];
    return(str);
}
/*****************************************************************
  OFFINT and DEFINT are subroutines which write the current
  force levels of this team's offensive and defensive forces
  into files so they can be read by the opponent's INTREP
  subroutine for intelligence gathering.                      */

offint(){
    intel = open("/usr/tempoblu/bluoff",2);
    for(i=0;i<=20;i++) write(intel,"                \n",14);
    close(intel);
    intel = open("/usr/tempoblu/bluoff",2);
    write(intel,"\n\nBLUE OFFENSIVE FORCE STATUS IS:\n",34);
    m = oa1[1];
    write(intel,"\nB-52G         ",13);
    message();
    m = oa1m[1];
    write(intel,"\nB-52G MOD     ",13);
    message();
    m = oa2[1];
    write(intel,"\nFB-111H       ",13);
    message();
    m = oa3[1];
    write(intel,"\nE-1           ",13);
    message();
```

```
    m = ob1[1];
    write(intel,"\nMM-III       ",13);
    message();
    m = ob1m[1];
    write(intel,"\nMM-III MOD   ",13);
    message();
    m = ob2[1];
    write(intel,"\nALCM         ",13);
    message();
    m = ob3[1];
    write(intel,"\nMX           ",13);
    message();
    m = ob4[1];
    write(intel,"\nFOB          ",13);
    message();
    close(intel);
}

defint(){
    intel = open("/usr/tempoblu/bludef",2);
    for(i=0;i<=20;i++) write(intel,"                \n",14);
    close(intel);
    intel = open("/usr/tempoblu/bludef",2);
    write(intel,"\n\nBLUE DEFENSIVE FORCE STATUS IS:\n",34);
    m = da1[1];
    write(intel,"\nCHAPPARAL    ",13);
    message();
    m = da2[1];
    write(intel,"\nHAWK         ",13);
    message();
    m = da3[1];
    write(intel,"\nPATRIOT      ",13);
    message();
    m = db1[1];
    write(intel,"\nSPRINT       ",13);
    message();
    m = db2[1];
```

```
    write(intel,"\nSPARTAN      ",13);
    message();
    m = db3[1];
    write(intel,"\nLoADS        ",13);
    message();
    m = db4[1];
    write(intel,"\nVADER        ",13);
    message();
    close(intel);
}
/*****************************************************************
  MESSAGE is a utility subroutine called by OFFINT and
  DEFINT to compute a ten-unit range around the number of
  actual force units the team has and then write that range
  into the intelligence report for the other side.          */

message(){
    char a[3],b[3];
    p = (m/10) * 10;
    q = p+9;
    x = p;
    itoas(x);
    write(intel,sstr,3);
    x = q;
    itoas(x);
    write(intel,"-",1);
    write(intel,sstr,3);
    return;
}
/*****************************************************************
  ITOAS is identical to ITOA except that it creates a three-
  character string instead of a five-character string.    */

itoas(){
    int ii,jj,kk;
    float yy;
    char rsstr[3];
```

```
    kk=0;
    while(kk<=2){
       yy = fmod(x,10.);
       ii = yy;
       rsstr[kk] = ii + '0';
       x = (x-yy)/10.;
       kk = kk+1;
    }
    for(jj=0;jj<=2;jj++) sstr[jj]=rsstr[2-jj];
    return(sstr);
}
/*****************************************************************
  ORDINT and DRDINT are subroutine which write the current
  active offensive and defensive R&D programs into file to
  be read by the opponent's INTREP routine.                    */

ordint(){
    intel = open("/usr/tempoblu/blurdo",2);
    for(i=0;i<=20;i++) write(intel,"                \n",14);
    close(intel);
    intel = open("/usr/tempoblu/blurdo",2);
    write(intel,"\n\nTHIS INTEL REPORT LISTS ALL OFFENSIVE\n",40);
    write(intel,"PROGRAMS FOR WHICH R&D FUNDS HAVE BEEN SPENT.\n",46);
    write(intel,"THIS YEAR.\n",11);
    if(tempflg[1]==1)
       write(intel,"\nB-52G MOD    ",13);
    if(tempflg[2]==1)
       write(intel,"\nFB-111H      ",13);
    if(tempflg[3]==1)
       write(intel,"\nB-1 BOMBER   ",13);
    if(tempflg[4]==1)
       write(intel,"\nMM-III MOD   ",13);
    if(tempflg[5]==1)
       write(intel,"\nALCM         ",13);
    if(tempflg[6]==1)
       write(intel,"\nMX ICBM      ",13);
    if(tempflg[7]==1)
```

```
      write(intel,"\nFOB          ",13);
if(tempflg[1]!=1&&tempflg[2]!=1&&tempflg[3]!=1&&tempflg[4]!=1
    &&tempflg[5]!=1&&tempflg[6]!=1&&tempflg[7]!=1)
      write(intel,"\nNONE         ",13);
   close(intel);
}

drdint(){
   intel = open("/usr/tempoblu/blurdd",2);
   for(i=0;i<=20;i++) write(intel,"             \n",14);
   close(intel);
   intel = open("/usr/tempoblu/blurdd",2);
   write(intel,"\n\nTHIS INTEL REPORT LISTS ALL DEFENSIVE\n",40);
   write(intel,"PROGRAMS FOR WHICH R&D FUNDS HAVE BEEN SPENT.\n",46);
   write(intel,"THIS YEAR.\n",11);
   if(tempflg[8]==1)
      write(intel,"\nHAWK  SAM    ",13);
   if(tempflg[9]==1)
      write(intel,"\nPATRIOT SAM  ",13);
   if(tempflg[10]==1)
      write(intel,"\nSPARTAN ABM  ",13);
   if(tempflg[11]==1)
      write(intel,"\nLoADS ABM    ",13);
   if(tempflg[12]==1)
      write(intel,"\nVADER ABM    ",13);
if(tempflg[8]!=1&&tempflg[9]!=1&&tempflg[10]!=1&&tempflg[11]!=1&&tempflg[12]!=1)
      write(intel,"\nNONE         ",13);
   close(intel);
}
/*************************************************************
  CNTRINT writes flags into a file to tell the opponent
  if this team has purchased intelligence information in
  any of four categories on him for this year.          */

cntrint(){
   int ci;
   ci= open("/usr/tempoblu/cntrint",2);
   x=intflag[1];
```

```
    itoas(x);
    write(ci,sstr,3);
    x=intflag[2];
    itoas(x);
    write(ci,sstr,3);
    x=intflag[3];
    itoas(x);
    write(ci,sstr,3);
    x=intflag[4];
    itoas(x);
    write(ci,sstr,3);
    close(ci);
}
/*****************************************************************
  YEAREND displays a message on the terminal to advise each
  team of a break period prior to the start of the next
  planning cycle.                                              */

yearend(){
   switch(year){
      case 1:
         printf("\n\n\nTHIS IS THE END OF YEAR 1. FOLLOWING A");
         if(mode==1)
            printf("\nFIVE MINUTE BREAK, YOU WILL BE GIVEN 20 MINUTES");
         else
            printf("\nFIFTEEN MINUTE BREAK, YOU WILL BE GIVEN 45 MINUTES");
            printf("\nTO PREPARE YOUR PLAN FOR YEAR 3. IF YOU REQUESTED\
                  \nINTELLIGENCE DATA YOU WILL RECEIVE IT AT THE \
                  \nBEGINNING OF NEXT YEAR'S PLAY.");
         break;

      case 2:
         printf("\n\n\nTHIS IS THE END OF YEAR 2. FOLLOWING A");
         if(mode==1)
            printf("\nFIVE MINUTE BREAK, YOU WILL BE GIVEN 15 MINUTES");
         else
            printf("\nTEN MINUTE BREAK, YOU WILL BE GIVEN 35 MINUTES");
```

```
            printf("\nTO PREPARE YOUR PLAN FOR YEAR 4. IF YOU REQUESTED\
                    \nINTELLIGENCE DATA YOU WILL RECEIVE IT AT THE \
                    \nBEGINNING OF NEXT YEAR'S PLAY.");
         break;

      case 3:
         printf("\n\n\nTHIS IS THE END OF YEAR 3. FOLLOWING A");
         if(mode==1)
            printf("\nFIVE MINUTE BREAK, YOU WILL BE GIVEN 10 MINUTES");
         else
            printf("\nTEN MINUTE BREAK, YOU WILL BE GIVEN 30 MINUTES");
            printf("\nTO PREPARE YOUR PLAN FOR YEAR 5. IF YOU REQUESTED\
                    \nINTELLIGENCE DATA YOU WILL RECEIVE IT AT THE \
                    \nBEGINNING OF NEXT YEAR'S PLAY.");
         break;

      case 4:
         printf("\n\n\nTHIS IS THE END OF YEAR 4. FOLLOWING A");
         if(mode==1)
            printf("\nFIVE MINUTE BREAK, YOU WILL BE GIVEN 20 MINUTES");
         else
            printf("\nTEN MINUTE BREAK, YOU WILL BE GIVEN 35 MINUTES");
            printf("\nTO PREPARE YOUR PLAN FOR YEAR 6-9. IF YOU REQUESTED\
                    \nINTELLIGENCE DATA YOU WILL RECEIVE IT AT THE\
                    \nBEGINNING OF NEXT YEAR'S PLAY.");
         break;

      case 5: case 6: case 7:
         printf("\n\n\nWAIT.\
               \nAFTER A MOMENTARY PAUSE YOU MUST CONTINUE.\
               \nYOUR CLOCK FOR THIS PLANNING YEAR IS STILL TICKING.");
         break;

      case 8:
         printf("\n\n\nTHE GAME IS FINISHED.  IN A FEW MINUTES\
                 \nYOU WILL RECEIVE A FINAL REPORT.");
         break;
```

```
        }
    return;
}

/*****************************************************************
**  This is the program ANALYZE.  It is called from the        **
**  BLUSIDE subroutine ANALYZE and it performs tradeoff        **
**  analyses between selected pairs of weapon systems.         **
**  This program is available only in years one through        **
**  four.                                                      **
*****************************************************************/
float t;
double fmod();
int i,acount,upay;
int oa1[11],oa1m[11],oa2[11],oa3[11],ob1[11],ob1m[11],ob2[11];
int ob3[11],ob4[11],da1[11],da2[11],da3[11],db1[11],db2[11],flag;
int db3[11],db4[11],w[11],x[11],y[11],xutil[11],yutil[11],year,yr;
int lcc[11],util[11],xlcc[11],ylcc[11],utltot,lcctot,xlt,ylt,xut,yut;
float utf,ltf,upd,xupd,yupd;
char clear 014,c1,c2,sys1,sys2,cont,sstr[3];
char str1[] "\n
\n              A - B-52G BOMBER\
\n              B - UPGRADED B-52G\
\n              C - FB-111H BOMBER\
\n              E - MINUTEMAN III ICBM\
\n              G - AIR LAUNCHED CRUISE MISSILE\
\n              J - CHAPPARAL SAM\
\n              M - SPRINT ABM\
\n              N - SPARTAN ABM";
char str2[] "\n
\n              A - B-52G BOMBER\
\n              B - UPGRADED B-52G\
\n              C - FB-111H BOMBER\
\n              E - MINUTEMAN III ICBM\
\n              F - UPGRADED MINUTEMAN\
\n              G - AIR LAUNCHED CRUISE MISSILE\
\n              H - MX MOBILE ICBM\
```

```
\n              J - CHAPPARAL SAM\
\n              K - IMPROVED HAWK SAM\
\n              M - SPRINT ABM\
\n              N - SPARTAN ABM\
\n              P - LoADS ABM";
cher str3[] "\n
\n              A - B-52G BOMBER\
\n              B - UPGRADED B-52G\
\n              C - FB-111H BOMBER\
\n              D - B-1 BOMBER\
\n              E - MINUTEMAN III ICBM\
\n              F - UPGRADED MINUTEMAN\
\n              G - AIR LAUNCHED CRUISE MISSILE\
\n              H - MX MOBILE ICBM\
\n              I - FOB SPACE-DIRECTED ICBM\
\n              J - CHAPPARAL SAM\
\n              K - IMPROVED HAWK SAM\
\n              L - PATRIOT SAM\
\n              M - SPRINT ABM\
\n              N - SPARTAN ABM\
\n              P - LoADS ABM\
\n              Q - VADER ABM";


main(argc,argv)
   int argc;
   int **argv;
   {
   int *ptr;
   ptr = *argv;
   year = *ptr;

   flag=0;
   acount=1;
   database();
   printf("%c\n\n\n\n",clear);
   printf("\n                    WELCOME TO THE ANALYSIS BRANCH.\
```

```
        \n\n              THIS SUBPROGRAM ENABLES YOU TO COMPARE\
        \n              TWO SYSTEMS TO DETERMINE WHICH IS THE\
        \n              MOST COST EFFECTIVE PURCHASE.\n\n");
    while(flag==0){
        printf("\nTHE CANDIDATE SYSTEMS FOR ANALYSIS WILL BE CHOSEN FROM THE\
          \nFOLLOWING LIST:");
        switch(year){
          case 1:
            printf("%s",str1);
            break;
          case 2:
            printf("%s",str2);
            break;
          case 3: case 4:
            printf("%s",str3);
            break;
          default:
            printf("\nIF YOU GET THIS ADVISORY SOMETHING IS WRONG!");
            break;
        }
printf("\n\nPLEASE SELECT THE LETTER OF THE FIRST SYSTEM IN THE COMPARISON.");
        select();
        sys1 = c1;
        for(i=1;i<11;i++) x[i]=w[i];
        printf("\n\nAND THE LETTER OF THE SECOND SYSTEM IN THE COMPARISON.");
        select();
        sys2 = c1;
        for(i=1;i<11;i++) y[i]=w[i];
        printf("%c\n",clear);
        printf("THE ");
        name(sys1);
        if(x[10]>0)
          rd(x);
        else
          oper(x);
        for(i=1;i<=10;i++) {xlcc[i]=lcc[i]; xutil[i]=util[i];}
        xlt=lcctot;
```

```
   xut=utltot;
   xupd=upd;

   printf("\n\nTHE ");
   name(sys2);
   if(y[10]>0)
      rd(y);
   else
      oper(y);
   for(i=year;i<=10;i++) {ylcc[i]=lcc[i]; yutil[i]=util[i];}
   ylt=lcctot;
   yut=utltot;
   yupd=upd;

   printf("\n\n                              ANALYSIS   YEAR %d ",year);
   printf("\n                    ");
   name(sys1);  printf(" vs ");  name(sys2);
printf("\n\nYEAR            COST PER YEAR                    UTILS PER YEAR");
printf("\n          ");name(sys1);printf("  ");name(sys2);printf("    ");
   name(sys1);printf("  ");name(sys2);

   yr = year-1;
   for(i=year;i<=10;i++){
      yr = yr+1;
      printf("\n  %d        ",yr);
      if(yr==10)
         printf("%d           %d",xlcc[i],ylcc[i]);
      else
         printf(" %d           %d",xlcc[i],ylcc[i]);
      if(xutil[i]==0&&yutil[i]==0)
         printf("              0000           0000");
      else if(xutil[i]==0&&yutil[i]!=0)
         printf("              0000           %d",yutil[i]);
      else
         printf("            %d0          %d0",xutil[i],yutil[i]);
   }
   printf("\n-------------------------------------------------------");
```

```
      printf("\nTOTALS    %d0          %d0          %d0          \
%d0",xlt,ylt,xut,yut);
      printf("\n\n\nUTILITY PER DOLLAR FOR ");
      name(sys1);
      printf("   %f",xupd);
      printf("\n\nUTILITY PER DOLLAR FOR ");
      name(sys2);
      printf("   %f",yupd);
      printf("\n\n DO YOU WISH TO ANALYZE ANY OTHER SYSTEMS? (y/n)");
      cont = getans();
      if(cont=='n'||cont=='N'){flag=1;
         t=acount;
         itoas(t);
         upay= open("/usr/rounce/saved/analpay",2);
         write(upay,sstr,3);
         close(upay);
      }
      else acount= acount+1;
         printf("%c",clear);
   }
}
/*************************************************************
  NAME is a utility subroutine called by ANALYZE to print
  the name of the desired weapon system on the terminal screen.*/

name(xyz){
   if(xyz=='a'||xyz=='A') printf("B-52G          ");
   if(xyz=='b'||xyz=='B') printf("B-52G MOD      ");
   if(xyz=='c'||xyz=='C') printf("FB-111H        ");
   if(xyz=='d'||xyz=='D') printf("B-1            ");
   if(xyz=='e'||xyz=='E') printf("MM-III         ");
   if(xyz=='f'||xyz=='F') printf("MM-III MOD     ");
   if(xyz=='g'||xyz=='G') printf("ALCM           ");
   if(xyz=='h'||xyz=='H') printf("MX             ");
   if(xyz=='i'||xyz=='I') printf("FOB            ");
   if(xyz=='j'||xyz=='J') printf("CHAPPARAL SAM  ");
   if(xyz=='k'||xyz=='K') printf("HAWK SAM       ");
```

```
   if(xyz=='l'||xyz=='L') printf("PATRIOT SAM  ");
   if(xyz=='m'||xyz=='M') printf("SPRINT ABM   ");
   if(xyz=='n'||xyz=='N') printf("SPARTAN ABM  ");
   if(xyz=='p'||xyz=='P') printf("LoADS ABM    ");
   if(xyz=='q'||xyz=='Q') printf("VADER ABM    ");
}
/*****************************************************************
  The RD subroutine is called by ANALYZE when the system
  being considered is an R&D program and not yet in the
  inventory. The subroutine calculates life cycle cost and
  life cycle utility and computes a utility per dollar
  factor.                                                    */

rd(z)int z[10];{
   int j;
   printf(" IS STILL IN THE R&D PHASE.  THE COSTS USED FOR THIS\
\nANALYSIS WILL BE THE LATEST ESTIMATES.  INCLUDED IN THE ANALYSIS ARE\
\nTHREE BUYS OF 20 UNITS, ONE IN EACH OF THE FIRST THREE YEARS THE\
\nSYSTEM IS AVAILABLE.  IF YOU HAVE ALREADY BEGUN R&D, ONLY THE REMAINING\
\nCOSTS OF DEVELOPMENT WILL BE CONSIDERED SINCE OUR CONCERN HERE IS FOR\
\nTHE VALUE RECEIVED FROM CURRENT AND FUTURE AVAILABLE FUNDS.  THE SYSTEMS\
\nPURCHASED WILL BE OPERATED FOR THE REMAINDER OF THE TEN YEAR GAME.");
   if(z[10]==1){
      lcc[year]=z[6]+20*z[2];          util[year]=0;
      lcc[year+1]=20*z[2] + 20*z[3];   util[year+1]=20*z[4];
      lcc[year+2]=20*z[2] + 40*z[3];   util[year+2]=40*z[4];
      lcc[year+3]=60*z[3];             util[year+3]=60*z[4];
      j=year+4;
      for(i=j;i<=10;i++) {lcc[i]=60*z[3]; util[i]=60*z[4];}
   }
   if(z[10]==2){ lcc[year]=z[6];                    util[year]=0;
      lcc[year+1]=z[7] + 20*z[2];      util[year+1]=0;
      lcc[year+2]=20*z[2] + 20*z[3];   util[year+2]=20*z[4];
      lcc[year+3]=20*z[2] + 40*z[3];   util[year+3]=40*z[4];
      lcc[year+4]=60*z[3];             util[year+4]=60*z[4];
      j=year+5;
      for(i=j;i<=10;i++) {lcc[i]=60*z[3]; util[i]=60*z[4];}
```

```
    }
    if(z[10]==3){ lcc[year]=z[6];                    util[year]=0;
       lcc[year+1]=z[7];                  util[year+1]=0;
       lcc[year+2]=z[8] + 20*z[2];        util[year+2]=0;
       lcc[year+3]=20*z[2] + 20*z[3];     util[year+3]=20*z[4];
       lcc[year+4]=20*z[2] + 40*z[3];     util[year+4]=40*z[4];
       lcc[year+5]=60*z[3];               util[year+5]=60*z[4];
       j=year+6;
       for(i=j;i<=10;i++) {lcc[i]=60*z[3]; util[i]=60*z[4];}
    }
    for(i=year;i<=10;i++) util[i]=discount(util[i]);
    lcctot=0;  utltot=0;
    for(i=year;i<11;i++) {lcctot=lcctot + lcc[i]/10;  utltot=utltot + util[i];}
    utf=utltot*10.;
    ltf=lcctot*10.;
    upd = utf/ltf;
    return;
}
/*****************************************************************
  OPER performs the same type of calculations as RD but since
  the systems here are already operational, the R&D costs do
  not need to be included.                                      */

oper(z)int z[10];{
   int j;
   printf(" IS CURRENTLY OPERATIONAL. SINCE THE FOCUS OF THE GAME\
\nIS HOW TO BEST UTILIZE THE AVAILABLE BUDGET RESOURCES FOR THE REMAINDER\
\nOF THE GAME, SUNK COSTS(e.g.PAST R&D) ARE NOT FACTORED INTO THE ICC\
\nCOMPARISON. THREE BUYS OF 20 UNITS WILL BE MADE, ONE IN EACH OF THE NEXT\
\nTHREE YEARS.  THE SYSTEMS PURCHASED WILL BE OPERATED FOR THE REMAINDER\
\nOF THE TEN YEAR GAME.");
   lcc[year]=20*z[2];                 util[year]=0;
   lcc[year+1]=20*z[2]+20*z[3];       util[year+1]=20*z[4];
   lcc[year+2]=20*z[2]+40*z[3];       util[year+2]=40*z[4];
   j=year+3;
   for(i=j;i<=10;i++) {lcc[i]=60*z[3];  util[i]=60*z[4];}
   for(i=year;i<=10;i++) util[i]=discount(util[i]);
```

```
   lcctot=0;  utltot=0;
  for(i=year;i<=10;i++) {lcctot=lcctot + lcc[i]/10;  utltot=utltot + util[i];}
   utf=utltot*10.;
   ltf=lcctot*10.;
   upd = utf/ltf;
   return;
}
/*****************************************************************
  SELECT is a utility subroutine which assigns database
  values to a dummy variable to streamline the analysis in
  RD and OPER. SELECT only allows analysis on those systems
  which are currently available for consideration.          */

select(){
   int trig1;
   trig1=0;
   while(trig1==0){
      c1=getans();
      switch(c1){
         case'a':case'A':for(i=1;i<=10;i++) w[i]=oa1[i];
                         trig1 =1;
                         break;
         case'b':case'B':for(i=1;i<=10;i++) w[i]=oa1m[i];
                         trig1 =1;
                         break;
         case'c':case'C':for(i=1;i<=10;i++) w[i]=oa2[i];
                         trig1 =1;
                         break;
         case'd':case'D':for(i=1;i<=10;i++) w[i]=oa3[i];
                         if(year==1||year==2) note();
                         else trig1 =1;
                         break;
         case'e':case'E':for(i=1;i<=10;i++) w[i]=ob1[i];
                         trig1 =1;
                         break;
         case'f':case'F':for(i=1;i<=10;i++) w[i]=ob1m[i];
                         if(year==1) note();
```

```
                    else trig1 =1;
                    break;
case'g':case'G':for(i=1;i<=10;i++) w[i]=ob2[i];
                    trig1 =1;
                    break;
case'h':case'H':for(i=1;i<=10;i++) w[i]=ob3[i];
                    if(year==1) note();
                    else trig1 =1;
                    break;
case'i':case'I':for(i=1;i<=10;i++) w[i]=ob4[i];
                    if(year==1||year==2) note();
                    else trig1 =1;
                    break;
case'j':case'J':for(i=1;i<=10;i++) w[i]=da1[i];
                    trig1 =1;
                    break;
case'k':case'K':for(i=1;i<=10;i++) w[i]=da2[i];
                    if(year==1) note();
                    else trig1 =1;
                    break;
case'l':case'L':for(i=1;i<=10;i++) w[i]=da3[i];
                    if(year==1||year==2) note();
                    else trig1 =1;
                    break;
case'm':case'M':for(i=1;i<=10;i++) w[i]=db1[i];
                    trig1 =1;
                    break;
case'n':case'N':for(i=1;i<=10;i++) w[i]=db2[i];
                    trig1 =1;
                    break;
case'p':case'P':for(i=1;i<=10;i++) w[i]=db3[i];
                    if(year==1) note();
                    else trig1 =1;
                    break;
case'q':case'Q':for(i=1;i<=10;i++) w[i]=db4[i];
                    if(year==1||year==2) note();
                    else trig1 =1;
```

```
                        break;
        default:        note();
                        break;
          }
      }
   return;
}
/*****************************************************************
  NOTE is a utility subroutine called by SELECT to indicate
  an unacceptable input.                                        */

note(){
   printf("\nYOUR INPUT WAS NOT A VALID CHOICE. PLEASE REENTER.");
}
/*****************************************************************
  GETANS accepts keyboard entries and rejects extraneous
  inputs.                                                       */

getans(){
   char buf[20],k;
   int l;
   for(l=0;(k=getchar())!='\n';l++) buf[l]=k;
   k=buf[0];
   return(k);
}
/*****************************************************************
  DATABASE was described above in BLUSID1.                      */

database(){
        /*
                second parameter:acquisition cost per unit
                third parameter: operating cost per unit per year
                fourth parameter:utils per unit
                fifth parameter: purchase limit per year
                sixth parameter: first year R&D cost
                seventh parameter:second year R&D cost
                eighth parameter:third year R&D cost
```

```
        ninth parameter: modification cost per unit
        tenth parameter: number of years until available    */

switch(year){
        case 1:

oa1[2]=60;      oa1m[2]=150;    oa2[2]=200;     ob1[2]=80;
oa1[3]=40;      oa1m[3]=80;     oa2[3]=50;      ob1[3]=60;
oa1[4]=2;       oa1m[4]=8;      oa2[4]=20;      ob1[4]=4;
oa1[5]=30;      oa1m[5]=30;     oa2[5]=0;       ob1[5]=20;
oa1[6]=0;       oa1m[6]=200;    oa2[6]=500;     ob1[6]=0;
oa1[7]=0;       oa1m[7]=0;      oa2[7]=400;     ob1[7]=0;
oa1[8]=0;       oa1m[8]=0;      oa2[8]=700;     ob1[8]=0;
oa1[9]=90;      oa1m[9]=0;      oa2[9]=0;       ob1[9]=30;
oa1[10]=0;      oa1m[10]=1;     oa2[10]=3;      ob1[10]=0;

ob2[2]=100;     da1[2]=100;     db1[2]=50;      db2[2]=100;
ob2[3]=90;      da1[3]=60;      db1[3]=40;      db2[3]=30;
ob2[4]=25;      da1[4]=5;       db1[4]=2;       db2[4]=10;
ob2[5]=0;       da1[5]=30;      db1[5]=30;      db2[5]=0;
ob2[6]=400;     da1[6]=0;       db1[6]=0;       db2[6]=700;
ob2[7]=600;     da1[7]=0;       db1[7]=0;       db2[7]=700;
ob2[8]=400;     da1[8]=0;       db1[8]=0;       db2[8]=0;
ob2[9]=0;       da1[9]=0;       db1[9]=0;       db2[9]=0;
ob2[10]=3;      da1[10]=0;      db1[10]=0;      db2[10]=2;

oa3[1]=1;       ob3[1]=1;       ob4[1]=1;       ob1m[1]=1;
da2[1]=1;       da3[1]=1;       db3[1]=1;       db4[1]=1;

break;

        case 2:

oa1[2]=60;      oa1m[2]=150;    ob1[2]=80;
oa1[3]=40;      oa1m[3]=80;     ob1[3]=60;
oa1[4]=2;       oa1m[4]=8;      ob1[4]=4;
oa1[5]=30;      oa1m[5]=30;     ob1[5]=20;
```

```
oa1[6]=0;       oa1m[6]=200;    ob1[6]=0;
oa1[7]=0;       oa1m[7]=0;      ob1[7]=0;
oa1[8]=0;       oa1m[8]=0;      ob1[8]=0;
ca1[9]=90;      oa1m[9]=0;      ob1[9]=30;
oa1[10]=0;      oa1m[10]=1;     ob1[10]=0;

oa2[2]=150;     ob1m[2]=100;    ob2[2]=120;
oa2[3]= 50;     ob1m[3]= 60;    ob2[3]=100;
oa2[4]=12;      ob1m[4]= 7;     ob2[4]=18;
oa2[5]=000;     ob1m[5]= 20;    ob2[5]=000;
oa2[6]=900;     ob1m[6]=100;    ob2[6]=800;
oa2[7]=900;     ob1m[7]=000;    ob2[7]=400;
oa2[8]=000;     ob1m[8]=000;    ob2[8]=000;
oa2[9]=000;     ob1m[9]=000;    ob2[9]=000;
oa2[10]=2;      ob1m[10]=1;     ob2[9]=2;

da1[2]=100;     db1[2]=50;
da1[3]=60;      db1[3]=40;
da1[4]=5;       db1[4]=2;
da1[5]=30;      db1[5]=30;
da1[6]=0;       db1[6]=0;
da1[7]=0;       db1[7]=0;
da1[8]=0;       db1[8]=0;
da1[9]=0;       db1[9]=0;
da1[10]=0;      db1[10]=0;

ob3[2]=150;     da2[2]=160;     db2[2]= 90;     db3[2]=100;
ob3[3]=100;     da2[3]=140;     db2[3]= 60;     db3[3]=50;
ob3[4]=28;      da2[4]=30;      db2[4]=11;      db3[4]=20;
ob3[5]=000;     da2[5]=000;     db2[5]=000;     db3[5]=0;
ob3[6]=300;     da2[6]=300;     db2[6]=400;     db3[6]=1000;
ob3[7]=600;     da2[7]=1000;    db2[7]=000;     db3[7]=0;
ob3[8]=800;     da2[8]=1000;    db2[8]=000;     db3[8]=0;
ob3[9]=000;     da2[9]=000;     db2[9]=000;     db3[9]=0;
ob3[10]=3;      da2[10]=3;      db2[10]=1;      db3[10]=1;

oa3[1]=1;       ob4[1]=1;       da3[1]=1;       db4[1]=1;
```

```
break;

        case 3:

oa1[2]=60;      oa1m[2]=150;    ob1[2]=80;
oa1[3]=40;      oa1m[3]=80;     ob1[3]=60;
oa1[4]=2;       oa1m[4]=8;      ob1[4]=4;
oa1[5]=30;      oa1m[5]=30;     ob1[5]=20;
oa1[6]=0;       oa1m[6]=200;    ob1[6]=0;
oa1[7]=0;       oa1m[7]=0;      ob1[7]=0;
oa1[8]=0;       oa1m[8]=0;      ob1[8]=0;
oa1[9]=90;      oa1m[9]=0;      ob1[9]=30;
oa1[10]=0;      oa1m[10]=1;     ob1[10]=0;

ob1m[2]=100;
ob1m[3]= 60;
ob1m[4]= 7;
ob1m[5]= 20;
ob1m[6]=100;
ob1m[7]=000;
ob1m[8]=000;
ob1m[9]=000;
ob1m[10]=1;

oa2[2]=200;     oa3[2]=200;     ob2[2]=130;     ob3[2]=180;
oa2[3]=50 ;     oa3[3]=80;      ob2[3]=100;     ob3[3]=110;
oa2[4]=10;      oa3[4]=20;      ob2[4]=20;      ob3[4]=30;
oa2[5]=000;     oa3[5]=0;       ob2[5]=000;     ob3[5]=000;
oa2[6]=700;     oa3[6]=800;     ob2[6]=400;     ob3[6]=500;
oa2[7]=000;     oa3[7]=800;     ob2[7]=000;     ob3[7]=500;
oa2[8]=000;     oa3[8]=800;     ob2[8]=000;     ob3[8]=000;
oa2[9]=000;     oa3[9]=0;       ob2[9]=000;     ob3[9]=000;
oa2[10]=1;      oa3[10]=3;      ob2[10]=1;      ob3[10]=2;

ob4[2]=100;     da2[2]=160;     da3[2]=200;
ob4[3]=80;      da2[3]=140;     da3[3]=60 ;
```

```
ob4[4]=16;      da2[4]=35;      da3[4]=20;
ob4[5]=0;       da2[5]=000;     da3[5]=000;
ob4[6]=800;     da2[6]=800;     da3[6]=1000;
ob4[7]=600;     da2[7]=900;     da3[7]=1800;
ob4[8]=0;       da2[8]=000;     da3[8]=000;
ob4[9]=0;       da2[9]=000;     da3[9]=000;
ob4[10]=2;      da2[10]=2;      da3[10]=2;

da1[2]=100;     db1[2]=50;
da1[3]=60;      db1[3]=40;
da1[4]=5;       db1[4]=2;
da1[5]=30;      db1[5]=30;
da1[6]=0;       db1[6]=0;
da1[7]=0;       db1[7]=0;
da1[8]=0;       db1[8]=0;
da1[9]=0;       db1[9]=0;
da1[10]=0;      db1[10]=0;

db2[2]= 90;     db3[2]=100;
db2[3]= 60;     db3[3]=50;
db2[4]=11;      db3[4]=20;
db2[5]=000;     db3[5]=0;
db2[6]=400;     db3[6]=1000;
db2[7]=000;     db3[7]=0;
db2[8]=000;     db3[8]=0;
db2[9]=000;     db3[9]=0;
db2[10]=1;      db3[10]=1;

break;

        case 4:

oa1[2]=60;      oa1m[2]=150;    ob1[2]=80;
oa1[3]=40;      oa1m[3]=80;     ob1[3]=60;
oa1[4]=2;       oa1m[4]=8;      ob1[4]=4;
oa1[5]=30;      oa1m[5]=30;     ob1[5]=20;
ca1[6]=0;       ca1m[6]=200;    ob1[6]=0;
```

```
oa1[7]=0;        oa1m[7]=0;       ob1[7]=0;
oa1[8]=0;        oa1m[8]=0;       ob1[8]=0;
oa1[9]=90;       oa1m[9]=0;       ob1[9]=30;
oa1[10]=0;       oa1m[10]=1;      ob1[10]=0;

ob1m[2]=100;
ob1m[3]= 60;
ob1m[4]= 7;
ob1m[5]= 20;
ob1m[6]=100;
ob1m[7]=000;
ob1m[8]=000;
ob1m[9]=000;
ob1m[10]=1;

oa2[2]=200;      ob2[2]=130;
oa2[3]=50 ;      ob2[3]=100;
oa2[4]=10;       ob2[4]=20;
oa2[5]=000;      ob2[5]=000;
oa2[6]=700;      ob2[6]=400;
oa2[7]=000;      ob2[7]=000;
oa2[8]=000;      ob2[8]=000;
oa2[9]=000;      ob2[9]=000;
oa2[10]=1;       ob2[10]=1;

oa3[2]=225;
oa3[3]=80 ;
oa3[4]=24;
oa3[5]=000;
oa3[6]=1100;
oa3[7]=900;
oa3[8]=000;
oa3[9]=000;
ca3[10]=2 ;


ob4[2]=120;      ob3[2]=180;      da2[2]=150;      da3[2]=150;
```

```
        ob4[3]=70;      ob3[3]=120;     da2[3]=120;     da3[3]=50 ;
        ob4[4]=15;      ob3[4]=33;      da2[4]=40;      da3[4]=25;
        ob4[5]=0;       ob3[5]=000;     da2[5]=000;     da3[5]=000;
        ob4[6]=800;     ob3[6]=300;     da2[6]=800;     da3[6]=1500;
        ob4[7]=0;       ob3[7]=000;     da2[7]=000;     da3[7]=000;
        ob4[8]=0;       ob3[8]=000;     da2[8]=000;     da3[8]=0;
        ob4[9]=0;       ob3[9]=0;       da2[9]=0;       da3[9]=0;
        ob4[10]=1;      ob3[10]=1;      da2[10]=1;      da3[10]=1;

        da1[2]=100;     db1[2]=50;
        da1[3]=60;      db1[3]=40;
        da1[4]=5;       db1[4]=2;
        da1[5]=30;      db1[5]=30;
        da1[6]=0;       db1[6]=0;
        da1[7]=0;       db1[7]=0;
        da1[8]=0;       db1[8]=0;
        da1[9]=0;       db1[9]=0;
        da1[10]=0;      db1[10]=0;

        db2[2]= 90;     db3[2]=100;
        db2[3]= 60;     db3[3]=50;
        db2[4]=11;      db3[4]=20;
        db2[5]=000;     db3[5]=0;
        db2[6]=400;     db3[6]=1000;
        db2[7]=000;     db3[7]=0;
        db2[8]=000;     db3[8]=0;
        db2[9]=000;     db3[9]=0;
        db2[10]=1;      db3[10]=1;

        }
}
/*************************************************************
  ITOAS is a utility subroutine which converts integer values
  to character strings.                                      */

itoas(){
   int ii,jj,kk;
```

```
   float yy;
   char rsstr[3];
   kk=0;
   while(kk<=2){
      yy = fmod(t,10.);
      ii = yy;
      rsstr[kk] = ii + '0';
      t = (t-yy)/10.;
      kk = kk+1;
   }
   for(jj=0;jj<=2;jj++) sstr[jj]=rsstr[2-jj];
   return(sstr);
}
/*****************************************************************
  DISCOUNT performs a disccunt function on the utility of
  multiple weapon systems.  This discounting is described
  in the User's Manual.                                        */

discount(gross){
   float disct;
   int disc;
   if(gross<=200) disct=gross;
   if(gross>200 && gross<=300) disct=200.+(.9*(gross-200));
   if(gross>300 && gross<=400) disct=290.+(.8*(gross-300));
   if(gross>400 && gross<=500) disct=370.+(.7*(gross-400));
   if(gross>500 && gross<=600) disct=440.+(.6*(gross-500));
   if(gross>600 && gross<=700) disct=500.+(.5*(gross-600));
   if(gross>700 && gross<=800) disct=550.+(.4*(gross-700));
   if(gross>800 && gross<=900) disct=590.+(.3*(gross-800));
   if(gross>900 && gross<=1000) disct=620.+(.2*(gross-900));
   if(gross>1000 && gross<=1100) disct=640.+(.1*(gross-1000));
   if(gross>1100) disct=650.;
   disc=disct;
   return(disc);
}
/*****************************************************************
**  This is the UMPIRE program called by DONE at the end**
```

```
**  of the year. It reads the Red Force data and compiles*
**  a cumulative yearly summary of comparative utility  **
**  totals for both teams.  This program also determines**
**  whether war has broken out and, if so, sends messages*
**  to that effect to both teams.                        **
*************************************************************/

int rand(),srand();
int atoi();
float atof();
double fmod();
char str[5],rdata[5];
float x;
int m,p,q;

main(argc,argv)
   int argc;
   char **argv;
   {
   float randno,bigrand;
   int rcumutl,bcumutl,ryrrep,byrrep;
   int blusum,blusumi,endblu,redsum,redsumi,endred,theend;
   int blurep,redrep,umprep,umprept,warrep,warrep1,blufb,redfb;
   int a,i,boffai,boffbi,bdefai,bdefbi,byeari,redwar, bluwar;
   int roffai,roffbi,rdefai,rdefbi,ryeari,redwini,bluwini;
   int bluairi,redairi,blumisi,redmisi,blutoti,redtoti,lstyri,war;
   char bluairc[5],redairc[5],blumisc[5],redmisc[5],blutotc[5],redtotc[5];
   char blusumc[5],redsumc[5],blustr[5],redstr[5];
   char *boffac,*boffbc,*bdefac,*bdefbc,*byearc,*randc,*lstyr,redwinc[5];
   char roffac[5],roffbc[5],rdefac[5],rdefbc[5],ryearc[5],bluwinc[5];

   byearc = *argv;
   boffac = *++argv;
   boffbc = *++argv;
   bdefac = *++argv;
   bdefbc = *++argv;
   randc  = *++argv;
```

```
/*****************************************************************
  The character string values carried across the fork from
  BLUSID1 are converted back into integer values.          */

   byeari = atoi(byearc);
   boffai = atoi(boffac);
   boffbi = atoi(boffbc);
   bdefai = atoi(bdefac);
   bdefbi = atoi(bdefbc);
   bigrand= atof(randc);
/*****************************************************************
  The REDSIDE data is read from the appropriate file and
  converted from character to integer values.              */

   ryeari=0;
   while(ryeari<byeari){
      if(byeari<=5)
         redrep = open("/usr/umpire/red",2);
      else if(byeari==6)
         redrep = open("/usr/umpire/red6",2);
      else if(byeari==7)
         redrep = open("/usr/umpire/red7",2);
      else if(byeari==8)
         redrep = open("/usr/umpire/red8",2);
      read(redrep,rdata,5);
      ryeari = atoi(rdata);
      for(i=0;i<5;i++) ryearc[i]=rdata[i];
      read(redrep,rdata,5);
      roffai = atoi(rdata);
      for(i=0;i<5;i++) roffac[i]=rdata[i];
      read(redrep,rdata,5);
      roffbi = atoi(rdata);
      for(i=0;i<5;i++) roffbc[i]=rdata[i];
      read(redrep,rdata,5);
      rdefai = atoi(rdata);
      for(i=0;i<5;i++) rdefac[i]=rdata[i];
```

```
      read(redrep,rdata,5);
      rdefbi = atoi(rdata);
      for(i=0;i<5;i++) rdefbc[i]=rdata[i];
      close(redrep);
   }
/*****************************************************************
  This portion of the program computes and writes to the
  file UMP the cumulative yearly summary.                        */

   if(byeari==1){
      umprept = open("/usr/umpire/ump",1);
      close(umprept);
   }
   umprep = open("/usr/umpire/ump",1);
   if(byeari>1) seek(umprep,0,2);
   write(umprep,"\n\n\n\nTHIS IS THE END OF YEAR ",28);
   write(umprep,byearc,5);
   write(umprep,"\n\nAIRCRAFT\n\n",12);
   write(umprep,"BLUE OFFENSIVE UTILS  ",22);
   write(umprep,boffac,5);
   write(umprep,"     RED OFFENSIVE UTILS  ",24);
   write(umprep,roffac,5);
   write(umprep,"\nRED DEFENSIVE UTILS   ",23);
   write(umprep,rdefac,5);
   write(umprep,"     BLUE DEFENSIVE UTILS  ",25);
   write(umprep,bdefac,5);
   write(umprep,"\n\nNET BLUE AIRCRAFT     ",24);

   bluairi = boffai - rdefai;
   if(bluairi<0) bluairi=0;
   x = bluairi;
   itoa(x);
   write(umprep,str,5);

   write(umprep,"     NET RED AIRCRAFT     ",24);
   redairi = roffai - bdefai;
   if(redairi<0) redairi=0;
```

```
x = redairi;
itoa(x);
write(umprep,str,5);

write(umprep,"\n\nMISSILES\n\n",12);
write(umprep,"BLUE OFFENSIVE UTILS  ",22);
write(umprep,boffbc,5);
write(umprep,"    RED OFFENSIVE UTILS  ",24);
write(umprep,roffbc,5);
write(umprep,"\nRED DEFENSIVE UTILS   ",23);
write(umprep,rdefbc,5);
write(umprep,"    BLUE DEFENSIVE UTILS  ",25);
write(umprep,bdefbc,5);
write(umprep,"\n\nNET BLUE MISSILES     ",24);

blumisi = boffbi - rdefbi;
if(blumisi<0) blumisi=0;
x = blumisi;
itoa(x);
write(umprep,str,5);

write(umprep,"    NET RED MISSILES     ",24);
redmisi = roffbi - bdefbi;
if(redmisi<0) redmisi=0;
x = redmisi;
itoa(x);
write(umprep,str,5);

write(umprep,"\n\nBLUE NET OFFENSIVE TOTAL",26);
write(umprep,"   RED NET OFFENSIVE TOTAL\n",27);
write(umprep,"                          ",26);
blutoti = bluairi + blumisi;
x = blutoti;
itoa(x);
write(umprep,str,5);
write(umprep,"                          ",26);
redtoti = redairi + redmisi;
```

```
    x = redtoti;
    itoa(x);
    write(umprep,str,5);

    bluwini=0;
    redwini=0;
    if(blutoti>redtoti){
       bluwini = blutoti-redtoti;
       write(umprep,"\n\n\n\nBLUE WINS THIS YEAR BY ",27);
       x = bluwini;
       itoa(x);
       write(umprep,str,5);
       write(umprep," UTILS.",7);
       for(i=0;i<5;i++) bluwinc[i]=str[i];
    }
    else if(redtoti>blutoti){
       redwini = redtoti-blutoti;
       write(umprep,"\n\n\n\nRED WINS THIS YEAR BY ",26);
       x = redwini;
       itoa(x);
       write(umprep,str,5);
       write(umprep," UTILS.",7);
       for(i=0;i<5;i++) redwinc[i]=str[i];
    }
    else
       write(umprep,"\n\n\n\nTHIS YEAR WAS A DRAW.",25);
    close(umprep);
/*************************************************************
  The next section of code writes messages into files which
  tell the teams what the yearly results of the game are.
  No utility figures are reported, only whether they won or
  lost.                                                    */

    if(blutoti>redtoti){
       byrrep = open("/usr/umpire/byrrep",2);
       write(byrrep,"YOU WON THIS YEAR---",20);
       close(byrrep);
```

```
        ryrrep = open("/usr/umpire/ryrrep",2);
        write(ryrrep,"YOU LOST THIS YEAR--",20);
        close(ryrrep);
    }
    if(redtoti>blutoti){
        byrrep = open("/usr/umpire/byrrep",2);
        write(byrrep,"YOU LOST THIS YEAR--",20);
        close(byrrep);
        ryrrep = open("/usr/umpire/ryrrep",2);
        write(ryrrep,"YOU WON THIS YEAR---",20);
        close(ryrrep);
    }
    if(redtoti==blutoti){
        byrrep = open("/usr/umpire/byrrep",2);
        write(byrrep,"THIS YEAR WAS A DRAW",20);
        close(byrrep);
        ryrrep = open("/usr/umpire/ryrrep",2);
        write(ryrrep,"THIS YEAR WAS A DRAW",20);
        close(ryrrep);
    }
/*************************************************************
  The next section compiles a game total of utility for
  each side so the overall game winner can be determined
  at the end of the game.                                    */

   blusum = open("/usr/umpire/blusum",2);
   read(blusum,blusumc,5);
   close(blusum);
   blusumi=atoi(blusumc);
   endblu = blusumi + blutoti;
   x = endblu;
   itoa(x);
   if(byeari<8){
      blusum = open("/usr/umpire/blusum",2);
      write(blusum,str,5);
      close(blusum);
   }
```

```
   if(byeari==8){ for(i=0;i<=4;i++) blustr[i]=str[i];}
   redsum = open("/usr/umpire/redsum",2);
   read(redsum,redsumc,5);
   close(redsum);
   redsumi=atoi(redsumc);
   endred = redsumi + redtoti;
   x = endred;
   itoa(x);
   if(ryeari<8){
      redsum = open("/usr/umpire/redsum",2);
      write(redsum,str,5);
      close(redsum);
   }
   if(ryeari==8){ for(i=0;i<=4;i++) redstr[i]=str[i];}
/*************************************************
  The next section actually writes the final report.    */

   if(byeari==8){
      theend = open("/usr/umpire/lastrep",2);
 write(theend,"THE FINAL UTIL TOTALS FOR THIS TEMPO GAME ARE AS FOLLOWS:",57);
      write(theend,"\n\nTOTAL BLUE FORCE NET OFFENSIVE UTILS    ",41);
      write(theend,blustr,5);
      write(theend,"\n\nTOTAL RED  FORCE NET OFFENSIVE UTILS    ",41);
      write(theend,redstr,5);
      if(endblu>endred){
         write(theend,"\n\nTHE WINNER IS:",16);
         write(theend,"  BLUE FORCE!",13);
      }
      if(endred>endblu){
         write(theend,"\n\nTHE WINNER IS:",16);
         write(theend,"   RED FORCE!",13);
      }
      if(endred==endblu) write(theend,"\n\n***THIS GAME WAS A DRAW.***",29);
      close(theend);
      printf("\n");
   }
/*************************************************
```

```
The next section uses a random number routine to determine
if a war has occurred this year.                          */

 war=0;
 randno = bigrand/32737.;
 warrep = open("/usr/umpire/war",2);
 read(warrep,lstyr,5);
 close(warrep);
 lstyri = atoi(lstyr);

 switch(byeari){
    case 1:
       if(randno<=.05) war=1;
       break;
    case 2:
       if(lstyri==1){
          if(randno<=.05) war=1;
          break;}
       else{
          if(randno<=.15) war=1;
          break;}
    case 3:
       if(lstyri==1){
          if(randno<=.10) war=1;
          break;}
       else{
          if(randno<=.35) war=1;
          break;}
    case 4:
       if(lstyri==1){
          if(randno<=.15) war=1;
          break;}
       else{
          if(randno<=.45) war=1;
          break;}
    case 5:
       if(lstyri==1){
```

```
            if(randno<=.25) war=1;
            break;}
         else{
            if(randno<=.65) war=1;
            break;}
      default:
         if(randno<=.25) war=1;
         break;
   }
/*****************************************************************
  The next section writes a flag into the file WAR so the
  teams can determine if a war has erupted during the year.
                                                              */
   x=war;
   itoa(x);
   warrep1 = open("/usr/umpire/war",2);
   write(warrep1,str,5);
   close(warrep1);
/*****************************************************************
  This section writes the actual war message into the team
  files BLUWAR1 and REDWAR1 which are read and output at
  the team terminals if a war has occurred.

   if(war==1){
      if(bluwini>0){
         bluwar = open("/usr/temploblu/bluwar1",2);
         write(bluwar,"\n\n\n\n\n\n",6);
         write(bluwar,"          W          W  AAAA  RRRRR\n",34);
         write(bluwar,"          W    W     W A    A R   R\n",34);
         write(bluwar,"          W    W W   W A    A RRRRR\n",34);
         write(bluwar,"          W  W   W  W AAAAAA RRR  \n",34);
         write(bluwar,"           W W   W W  A    A R  R \n",34);
         write(bluwar,"            W    W   A    A R   R\n",34);
         write(bluwar,"\n\n\n           HAS BROKEN OUT THIS YEAR!!",39);
         write(bluwar,"\n\n\n\n IT WAS A SHORT-LIVED CONFLICT AND ",39);
         write(bluwar,"YOUR FORCES WERE VICTORIOUS.\n",29);
         write(bluwar,"THE PRICE OF YOUR SUCCESS IS A $400",35);
```

```
write(bluwar,"  REDUCTION IN YOUR NEXT YEAR'S\n",31);
write(bluwar,"BUDGET. YOUR OPPONENT DID NOT",29);
write(bluwar,"  FARE NEARLY AS WELL.**************\n",37);
write(bluwar,"**********************************",32);
write(bluwar,"**********************************\n",34);
write(bluwar,"**********************************",32);
write(bluwar,"**********************************\n",34);
write(bluwar,"**********************************",32);
write(bluwar,"**********************************\n",34);
close(bluwar);

x=400.;
itoa(x);
blufb = open("/usr/tempoblu/blufdbk",2);
write(blufb,str,5);
close(blufb);

redwar = open("/usr/tempored/redwar1",2);
write(redwar,"\n\n\n\n\n\n",6);
write(redwar,"          W          W  AAAA  RRRRR\n",34);
write(redwar,"          W    W     W A    A R   R\n",34);
write(redwar,"          W   W W    W A    A RRRRR\n",34);
write(redwar,"          W  W   W   W AAAAAA RRR  \n",34);
write(redwar,"           W W    W W  A    A R  R \n",34);
write(redwar,"            W      W   A    A R   R\n",34);
write(redwar,"\n\n\n           HAS BROKEN OUT THIS YEAR!!",39);
write(redwar,"\n\n\n\n IT WAS A SHORT-LIVED CONFLICT AND ",39);
write(redwar,"YOUR FORCES WERE DEFEATED...\n",29);
write(redwar,"THE CONSEQUENCES OF THIS CONFLICT ARE",37);
write(redwar,"  A $400 REDUCTION IN YOUR NEXT\n",31);
write(redwar,"YEAR'S BUDGET PLUS A FURTHER  ",29);
write(redwar,"BUDGET REDUCTION AMOUNTING TO\n",30);
write(redwar,"TWICE THE DIFFERENCE BETWEEN  ",29);
write(redwar,"THE NET OFFENSIVE UTIL TOTALS\n",30);
write(redwar,"OF YOU AND YOUR OPPONENT. THIS  ",31);
write(redwar,"COMES TO A TOTAL BUDGET REDUCTION\n",34);
write(redwar,"FOR NEXT YEAR OF $",18);
```

```
    a = (2*bluwini)+400;
    x=a;
    itoa(x);
    write(redwar,str,5);
    write(redwar," (EXCLUDING ANY OTHER PENALTIES YOU\n",36);
    write(redwar,"MAY HAVE ACCUMULATED.",21);
    close(redwar);

    redfb = open("/usr/tempored/redfdbk",2);
    write(redfb,str,5);
    close(redfb);
}
else if(redwini>0){
    redwar = open("/usr/tempored/redwar1",2);
    write(redwar,"\n\n\n\n\n\n\n",6);
    write(redwar,"          W          W  AAAA  RRRRR\n",34);
    write(redwar,"          W    W     W A    A R   R\n",34);
    write(redwar,"          W  W W     W A    A RRRRR\n",34);
    write(redwar,"          W  W   W  W AAAAAA RRR  \n",34);
    write(redwar,"           W W    W W  A    A R  R \n",34);
    write(redwar,"            W      W   A    A R   R\n",34);
    write(redwar,"\n\n\n           HAS BROKEN OUT THIS YEAR!!",39);
    write(redwar,"\n\n\n\n IT WAS A SHORT-LIVED CONFLICT AND ",39);
    write(redwar,"YOUR FORCES WERE VICTORIOUS.\n",29);
    write(redwar,"THE PRICE OF YOUR SUCCESS IS A $400",35);
    write(redwar," REDUCTION IN YOUR NEXT YEAR'S\n",31);
    write(redwar,"BUDGET. YOUR OPPONENT DID NOT",29);
    write(redwar," FARE NEARLY AS WELL.**********\n",37);
    write(redwar,"**********",32);
    write(redwar,"**********\n",34);
    write(redwar,"**********",32);
    write(redwar,"**********\n",34);
    write(redwar,"**********",32);
    write(redwar,"**********\n",34);
    close(redwar);

    x=400.;
```

```
    itoa(x);
    redfb = open("/usr/tempored/redfdbk",2);
    write(redfb,str,5);
    close(redfb);

    bluwar = open("/usr/tempoblu/bluwar1",2);
    write(bluwar,"\n\n\n\n\n\n",6);
    write(bluwar,"          W          W  AAAA  RRRRR\n",34);
    write(bluwar,"          W     W    W A    A R    R\n",34);
    write(bluwar,"          W    W W   W A    A RRRRR\n",34);
    write(bluwar,"          W  W   W  W AAAAAA RRR  \n",34);
    write(bluwar,"           W W   W W  A    A R  R \n",34);
    write(bluwar,"            W     W   A    A R   R\n",34);
    write(bluwar,"\n\n\n           HAS BROKEN OUT THIS YEAR!!",39);
    write(bluwar,"\n\n\n\n IT WAS A SHORT-LIVED CONFLICT AND ",39);
    write(bluwar,"YOUR FORCES WERE DEFEATED.\n",27);
    write(bluwar,"THE CONSEQUENCES OF THIS CONFLICT ARE",37);
    write(bluwar," A $400 REDUCTION IN YOUR NEXT\n",31);
    write(bluwar,"YEAR'S BUDGET PLUS A FURTHER ",29);
    write(bluwar,"BUDGET REDUCTION AMOUNTING TO\n",30);
    write(bluwar,"TWICE THE DIFFERENCE BETWEEN ",29);
    write(bluwar,"THE NET OFFENSIVE UTIL TOTALS\n",30);
    write(bluwar,"OF YOU AND YOUR OPPONENT. THIS ",31);
    write(bluwar,"COMES TO A TOTAL BUDGET REDUCTION\n",34);
    write(bluwar,"FOR NEXT YEAR OF $",18);
    a = (2*redwini)+400;
    x=a;
    itoa(x);
    write(bluwar,str,5);
    write(bluwar," (EXCLUDING ANY OTHER PENALTIES YOU\n",36);
    write(bluwar,"MAY HAVE ACCUMULATED.",21);
    close(bluwar);

    blufb = open("/usr/tempoblu/blufdbk",2);
    write(blufb,str,5);
    close(blufb);
}
```

```
else{
   redwar = open("/usr/tempored/redwar1",2);
   write(redwar,"\n\n\n\n\n\n\n",6);
   write(redwar,"          W          W  AAAA  RRRRR\n",34);
   write(redwar,"          W     W    W A    A R   R\n",34);
   write(redwar,"          W   W W    W A    A RRRRR\n",34);
   write(redwar,"          W  W   W   W AAAAAA RRR  \n",34);
   write(redwar,"           W W   W W  A    A R  R \n",34);
   write(redwar,"            W     W   A    A R   R\n",34);
   write(redwar,"\n\n\n           HAS BROKEN OUT THIS YEAR!!",39);
   write(redwar,"\n\n\n\n IT WAS A SHORT-LIVED CONFLICT AND ",39);
   write(redwar,"YOU FOUGHT TO A STANDSTILL..\n",29);
   write(redwar,"WITH YOUR OPPONENT. THE COST OF THIS",36);
   write(redwar," ENCOUNTER WILL BE A $400  \n",28);
   write(redwar,"REDUCTION IN YOUR BUDGET FOR NEXT YEAR.",39);
   write(redwar,"**************************\n",27);
   write(redwar,"**********************************",32);
   write(redwar,"**********************************\n",34);
   write(redwar,"**********************************",32);
   write(redwar,"**********************************\n",34);
   write(redwar,"**********************************",32);
   write(redwar,"**********************************\n",34);
   close(redwar);

   x=400.;
   itoa(x);
   redfb = open("/usr/tempored/redfdbk",2);
   write(redfb,str,5);
   close(redfb);

   bluwar = open("/usr/tempoblu/bluwar1",2);
   write(bluwar,"\n\n\n\n\n\n\n",6);
   write(bluwar,"          W          W  AAAA  RRRRR\n",34);
   write(bluwar,"          W     W    W A    A R   R\n",34);
   write(bluwar,"          W   W W    W A    A RRRRR\n",34);
   write(bluwar,"          W  W   W   W AAAAAA RRR  \n",34);
   write(bluwar,"           W W   W W  A    A R  R \n",34);
```

```
        write(bluwar,"          W     W   A    A R   R\n",34);
        write(bluwar,"\n\n\n          HAS BROKEN OUT THIS YEAR!!",39);
        write(bluwar,"\n\n\n\n IT WAS A SHORT-LIVED CONFLICT AND ",39);
        write(bluwar,"YOU FOUGHT TO A STANDSTILL..\n",29);
        write(bluwar,"WITH YOUR OPPONENT. THE COST OF THIS",36);
        write(bluwar," ENCOUNTER WILL BE A $400  \n",28);
        write(bluwar,"REDUCTION IN YOUR BUDGET FOR NEXT YEAR.",39);
        write(bluwar,"**************************\n",27);
        write(bluwar,"*********************************",32);
        write(bluwar,"*********************************\n",34);
        write(bluwar,"*********************************",32);
        write(bluwar,"*********************************\n",34);
        write(bluwar,"*********************************",32);
        write(bluwar,"*********************************\n",34);
        close(bluwar);

        x=400.;
        itoa(x);
        blufb = open("/usr/tempoblu/blufdbk",2);
        write(blufb,str,5);
        close(blufb);
    }
  }
}
/*****************************************************************
  ITOA is the standard integer to ascii conversion routine
  used in the other program segments.                          */

itoa(){
   int i,j,k;
   float y;
   char rstr[5];
   k=0;
   while(k<=4){
      y = fmod(x,10.);
      i = y;
      rstr[k] = i + '0';
```

```
        x = (x-y)/10.;
        k = k+1;
    }
    for(j=0;j<=4;j++) str[j]=rstr[4-j];
    return(str);
}
```

# BIBLIOGRAPEY

Air University, Squadron Officers' School Management Text, Area 3, Book 2, p. 3230-S-1 through 3230-S-30, 1979.

Air University, Squadron Officers' School, TEMPO Faculty Lesson Plan, p. 3230-P-1 through 3230-P-28, 1979.

Hatry, H., Jackson, F., and Lever, P., TEMPO Military Planning Game - Rules and Suggestions for Players, working papers obtained from Professor A. Andrus, August 1980.

INITIAL DISTRIBUTION LIST

| | | No. Copies |
|---|---|---|
| 1. | Superintendent<br>Naval Postgraduate School<br>ATTN: Code 0142<br>Monterey, California 93940 | 2 |
| 2. | Defense Technical Information Center<br>Cameron Station<br>Alexandria, Virginia 22314 | 2 |
| 3. | Superintendent<br>Naval Postgraduate School<br>ATTN: Code 74<br>Monterey, California 93940 | 3 |
| 4. | Superintendent<br>Naval Postgraduate School<br>ATTN: Code 39<br>Monterey, California 93940 | 1 |
| 5. | Superintendent<br>Naval Postgraduate School<br>ATTN: Code 55Rh<br>Monterey, California 93940 | 1 |
| 6. | Superintendent<br>Naval Postgraduate School<br>ATTN: Code 55As<br>Monterey, California 93940 | 1 |
| 7. | Superintendent<br>Naval Postgraduate School<br>ATTN: Code 52<br>Monterey, California 93940 | 1 |
| 8. | AFIT/CISP<br>ATTN: Major Charles Earnhart, USAF<br>Wright Patterson Air Force Base, Ohio 45433 | 1 |
| 9. | HQ AU/ACDY<br>ATTN: Captain Summer<br>Building 744<br>Maxwell Air Force Base, Alabama 36112 | 1 |

10. SOS/EDCK 1
ATTN: Captain McBride
Maxwell Air Force Base, Alabama 36112

11. Lt Col Thomas P. Stack, USAF 1
SHAPE/SHOC
APO New York, New York 09055

12. Captain Scott Rounce 3
24866 Apollo
Mission Viejo, California 92691

Thesis
R787
c.1

192385

Rounce
An interactive military planning game for the Naval Postgraduate School Command, Control, and Communications Laboratory.

9 NOV 82 130491
12 NOV 86 30822
21 OCT 88 35122

Thesis
R787
c.1

192385

Rounce
An interactive military planning game for the Naval Postgraduate School Command, Control, and Communications Laboratory.

thesR787
An interactive military planning game fo
3 2768 000 99999 9
DUDLEY KNOX LIBRARY